robotron

# Anwenderdokumentation

# Sprachbeschreibung für Geräte mit dem Betriebssystem SIOS 1520

# Assembler-Sprachbeschreibung MABS 1520

# Beschreibung der Makrobefehle

Inhaltsverzeichnis

1. Charakteristik von MABS 1520
2. Metasprache für die Syntaxdarstellung
3. Format der Sprache MABS 1520
4. Sprachelemente
5. Beschreibung der MABS 1520-Pseudobefehle
6. Datenarten
7. Assemblerlistenausgabe
8. Beschreibung der Makrobefehle

Anlagen 1-4

## 1.
## Charakteristika von MABS 1520

Die Sprache MABS 1520 ist eine maschinenorientierte symbolische Assemblerprogrammiersprache (MABS - Makro-Befehls-Sprache).

Die Makrobefehle sind Maschinenbefehle einer höheren Ebene (über CPU-Befehle-Ebene) und werden durch den Makrointerpreter MINT 1520 interpretativ abgearbeitet. Weiterhin bietet MABS 1520 eine Reihe von Pseudobefehlen, die der Steuerung der Arbeit des Assemblers dienen.

Der Befehlsumfang ist auf das K 1520-System mit dem Betriebssystem SIOS 1520 zugeschnitten. Damit werden die Belange der Datenerfassungs-, Buchungs- und Fakturiertechnik sowie der Verarbeitung berücksichtigt.

Die Übersetzung in MABS 1520 geschriebener Programme erfolgt mit dem Übersetzungsprogramm MBAS 1520 (SIEX) (MBAS = Makro-Befehls-Assembler).

Symbolische Programme sind leichter lesbar und übersichtlicher als Programme im Maschinencode. Ihre Erstellungs-, Test- und Änderungszeit ist kürzer, da bestimmte Fehlerquellen vermieden und bei der Übersetzung formale Sprachfehler vom MBAS 1520 (SIEX) erkannt werden.

Die sprachlichen Mittel gestatten bei symbolisch geschriebenen Programmen, verschiebbare Programme zu erstellen sowie für einzeln erstellte Programmteile, Programme und Unterprogramme die logischen Verbindungen untereinander zu definieren. Die Verbindungsdefinitionen werden vom Binder ausgewertet und ermöglichen das physische Verbinden (Zusammenfügen) der Programme.

## 2.
## Metasprache für die Syntaxdarstellung

Zur Darstellung der Sprachsyntax wird eine Metasprache benutzt, die nach folgenden Regeln aufgebaut ist:

1. Worte der Metasprache werden klein geschrieben
2. In der Metasprache finden folgende Zeichen mit definierter Bedeutung Anwendung:

| Zeichen der Metasprache | Bedeutung |
|---|---|
| [ ] | Ein in eckige Klammern eingeschlossenes Element der Sprache kann fehlen |
| ::= | "besteht aus" |
| \| oder { } | "\|" oder "{ }" sind Auswahlzeichen, d.h. sie trennen gleichberechtigte Elemente (Funktion: oder) |
| $[\ ]^{*i}$ | Das in eckige Klammern eingeschlossene Element kann i mal auftreten |
| := | "ergibt sich zu" |

## 3. Format der Sprache MABS 1520

Ein Quellprogramm besteht aus einer Folge von Quellzeilen. Die maximale Länge einer Quellzeile beträgt 60 Zeichen. Quellzeilen werden mit dem Editor (EDIT 1520 (SIEX)) erfaßt. Der Editor generiert automatisch das Zeilenende (NL).

Eine Quellzeile kann sein

eine Befehlszeile oder
eine Kommentarzeile oder
eine Folgezeile (nur nach DDB-Befehlszeile!).

### 3.1. Befehlszeile

Eine Quellzeile, die einen symbolischen Befehl enthält, besteht aus 4 Feldern variabler Länge. Sie werden durch Trennzeichen (t 1, t 2, t 3) voneinander getrennt:

| | | | |
|---|---|---|---|
| Marken- t 1 feld | Operations- t 2 codefeld | Operanden- t 3 feld | Kommen- NL tarfeld |

Die Quellzeile wird immer durch das Zeilenendezeichen (NL) abgeschlossen (NL wird automatisch vom Editor generiert).

#### Markenfeld

Das Markenfeld kann maximal 5 Positionen umfassen. Es wird benutzt, um einem Befehl oder einer Konstanten bzw. einem Speicherbereich eine symbolische Adresse (Symbol) zuzuweisen. Damit kann auf diese symbolische Adresse von jeder beliebigen Programmstelle Bezug genommen werden.

Die Symbole bestehen aus 1 bis 5 Buchstaben oder Ziffern, wobei das 1. Zeichen ein Buchstabe sein muß.

Das Markenfeld wird durch das Trennzeichen ':' abgeschlossen. Das Trennzeichen folgt unmittelbar auf die letzte Stelle des Symbols. Das Zeichen ':' gibt an, daß das Symbol beim Übersetzen als Marke zu werten ist.

Die Zuweisung der Speicherplätze zu den symbolischen Adressen erfolgt bei der Übersetzung.

Operationscodefeld (OP-Codefeld)

Das Operationscodefeld enthält die Befehlsmnemonik (Operationscode der symbolischen Anweisung). Sämtliche Makrobefehlsmnemoniks bestehen aus 1 bis 5 Zeichen, sämtliche Mnemoniks der Pseudobefehle aus 2 bis 5 Zeichen.

Dieses Feld weist die Maschinenoperation bzw. Pseudooperation an, die auszuführen ist.

Das Operationscodefeld wird vom Operandenfeld durch mindestens ein Trennzeichen 'space' getrennt.

Operandenfeld

Das Operandenfeld dient der Aufnahme der Operanden (Operandenliste), die vom Typ des Befehls abhängen (s. Operandentypen Abschnitt 4 und Gruppeneinteilung der Befehls nach Operandentypen in Anlage 1).

Das Operandenfeld endet mit 'zeilenende', wenn kein Kommentar folgt bzw. mit '[space];', falls Kommentar folgt.

Kommentarfeld

Kommentar erleichtert dem Programmierer das Verständnis und die Wartung des Programms.

Ein wahlweise angefügter Kommentar hat mit einem Semikolon (';') zu beginnen und endet mit 'zeilenende'.

Nur das OP-Codefeld ist in einer Befehlszeile unbedingt anzugeben.

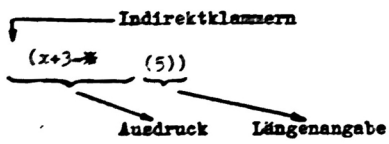
Formaler Aufbau einer Befehlszeile:

[symbol] [space] op-code [space] [operandenliste] [space] [kommentar] zeilenende

space ::= b | space (d.h. mehrere b sind zulässig)
zeilenende ::= NL (H'ØD')

Beim Schreiben des Operandenfeldes eines symbolischen Befehls sind folgende syntaktische Regeln zu beachten:

1. Das Operandenfeld ist vom OP-Codefeld mindestens durch ein Leerzeichen (b) zu trennen.
   Abgeschlossen wird es durch [space] zeilenende bzw. (wenn Kommentar folgt) durch [space]; kommentar zeilenende.
2. Operanden untereinander werden durch Komma getrennt. Leerzeichen (b) zwischen den Operanden sind verboten.
3. Leerzeichen innerhalb der Operanden sind nur bei $zkc_1$ zugelassen (s. Pkt. 4.5.3.).
4. Längenangaben, die zu Ausdrücken gehören, werden in runde Klammern '()' eingeschlossen.
5. Wird eine Indirektadressierung vorgenommen, ist der Ausdruck einschließlich einer möglichen Längenangabe in Klammern einzuschließen,
   z.B.



## 3.2. Kommentarzeile

Eine ganze Zeile gilt als Kommenterzeile, wenn sie mit ';' bzw. mit [space]';' beginnt.

Formaler Aufbau der Kommentarzeile:

[space]; kommentar zeilenende

space ::= ␣ | ␣space (d.h. mehrere ␣ sind zulässig)
zeilenende ::= NL (H'ØD')

Die Kommentare haben keinen Einfluß auf die Übersetzung.

## 3.3. Folgezeile

Eine mit '*' oder 'space *' begonnene Zeile wird "Folgezeile" genannt und ist nur nach der Pseudoanweisung DDB zulässig (s. Pkt. 5.12.). Sie enthält weitere Parameter der Parameterliste der DDB-Anweisung, die nicht in der Befehlszeile untergebracht werden konnten.

Formaler Aufbau der Folgezeile:

[space] * operandenliste [;kommentar] zeilenende

space ::= ␣ | ␣space (d.h. mehrere ␣ sind zulässig)
zeilenende ::= NL (H'ØD')

## 3.4. Quellprogrammbeispiel

Das folgende Beispiel zeigt den Aufbau eines Quellprogramms.

```
          PN        ABC
; ***********************************
; *EINGABE + DRUCK NUM.WERTE MIT L=6 *
; ***********************************

          SOP
MA123 ;   POS   11    ,H=1Ø,V+1       ; DRUCKERPOSITIONIERUNG
          ENT   FELD(6)               ; NUM. EINGABE
          PUT   11    ,FELD(6), Ø     ; DRUCK
          B     MA123
FELD:     DS    6                     ; FELD DER LAENGE 6
          END
```

## 4.
## Sprachelemente

### 4.1.
### Zeichen

Für MABS 1520 gelten die Codefestlegungen nach RGW 356-76, 7-Bit-Code (KOI-7).

Einige Zeichen haben eine besondere Bedeutung in der MABS 1520-Sprache:

Elemente der Sprache mit besonderer Bedeutung:

| Zeichen | Bedeutung |
|---|---|
| : | Im Markenfeld in Verbindung mit einem Symbol als Kennzeichnung der Marke und damit Trennzeichen zwischen Marken- und OP-Codefeld |
| * | Dreifache Bedeutung:<br>1. Steht * als erstes Zeichen einer Quellzeile, so wird diese als Folgezeile definiert (s. DDB-Pseudobefehl Abschnitt 5.12.).<br>2. Im Operandenfeld innerhalb eines Ausdrucks gibt * den laufenden Stand des Speicherplatzzuweisungszählers an.<br>3. Trennzeichen zwischen 2 Ausdrücken in erweiterter DS-Anweisung (ist wie "mal" zu lesen) |
| + - | Operatoren in Ausdrücken (s. Abschnitt 4.8.) |
| , (Komma) | Trennungszeichen für Ausdrücke in der Operandenliste des Operandenfeldes |

| Zeichen | Bedeutung |
|---|---|
| ; | 1. Als erstes Zeichen einer Quellzeile kennzeichnet es eine Kommentarzeile.<br>2. Wird es im Kommentarfeld angegeben, dient es als Trennungszeichen zwischen Operanden- und Kommentarfeld. |
| ( ) | Doppelte Bedeutung:<br>1. Kennzeichnung des Längenattributs des 1. Elements eines Ausdrucks<br>2. Kennzeichnung der Indirektadressierung (adr)<br>(adr(längenattribut)) |
| = | Spezifikation eines Parameters |
| ' (Apostroph) | Kennzeichnung einer Zeichenkette |

4.1.1.
Alphanumerische Zeichen ($c_i$)

Es sind alle alphanumerischen Zeichen entsprechend RGW 356-76, 7-Bit-Code, zugelassen (Spalten 2 - 5 der Codetabelle).

$$c_i \in \{@, A, \ldots, Z, Ø, \ldots, 9, Ü, !\ldots\}$$

Beachte:
Kleinbuchstaben sind nicht zugelassen.

4.1.2.
Hexadezimalziffern ($h_i$)

Eine Hexadezimalziffer (auch Hexaziffer) ist eines der nachfolgenden Zeichen:

$$\text{Hexaziffer} \in \{Ø, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F\}$$

## 4.2. Mnemoniks

Die symbolischen Abkürzungen der Makro- und Pseudobefehle werden als Mnemoniks (bzw. Befehlsmnemoniks) bezeichnet. (Mnemoniks der Makrobefehle können 1 bis 5 Zeichen, Mnemoniks der Pseudobefehle 2 bis 5 Zeichen umfassen.)
Die von MABS 1520 für die Makrobefehle zugelassenen Mnemoniks werden in Anlage 2 angegeben. Eine Beschreibung der Pseudobefehle und die Angabe ihrer Mnemoniks erfolgt unter Pkt. 5.

## 4.3. Symbol

Ein Symbol besteht aus maximal 5 Zeichen, beginnend mit einem Buchstaben. Weitere Zeichen können Buchstaben oder Ziffern sein.

Sonderzeichen sind nicht zugelassen.

Symbole können sowohl im Markenfeld als auch im Operandenfeld stehen. Ein Symbol verkörpert einen Adreßwert und ein Längenattribut. Das Adreßattribut eines Symbols im Markenfeld ist gleich dem aktuellen Wert des Speicherplatzzuweisungszählers. Ist der markierte Befehl (Symbol im Markenfeld) ein Makrobefehl, so ist das Längenattribut gleich 1. Ist der markierte Befehl eine Konstantendefinition oder eine Speicherplatzvereinbarung, so ist das Längenattribut gleich der Länge der Konstanten oder des reservierten Speicherbereichs.

Ausnahmen bilden die durch den Pseudobefehl EQU (Equate) oder die erweiterte DS-Anweisung definierten Symbole. Im Falle einer EQU-Anweisung erhält das Symbol den Adreßwert des im Operandenfeld stehenden Ausdrucks und dessen Längenattribut und im Falle der erweiterten DS-Anweisung als Längenattribut den Wert des 2. Ausdrucks (falls Wert $>256$, so LA = 1).

Jedes Symbol, das im Programm im Markenfeld auftritt, wird als definiert bezeichnet.

Tritt einunddasselbe Symbol ein zweites Mal im Markenfeld auf, so erfolgt bei der Übersetzung mit MBAS 1520 (SIEX) ein Fehlerausdruck (Anzeige). Gleichfalls werden nicht definierte Symbole, die im Operandenfeld auftreten, durch einen Fehlerausdruck (Anzeige) gekennzeichnet.

Kein Adreßwert kann zugeordnet werden, wenn das Symbol als externes gekennzeichnet ist (s. Pseudoanweisung EXTRN, Abschnitt 5.9.), so ist dies im obigen Sinne auch ein definiertes Symbol. Die Wertzuweisung erfolgt im Bindelauf mit dem Systemprogramm LINK 1520 (SIEX).

Beispiel für Symbole:

| Zulässige Symbole | Unzulässige Symbole |
|---|---|
| A1 | 1 |
| MARKE | MARKEX |
| X128 | 128X |
| | MARK# |
| | A + B |

## 4.4. Bezugnahme auf den Speicherplatzzuweisungszähler (*)

Durch * im Operanden wird sich im Programm auf den aktuellen Wert des Speicherplatzzuweisungszählers bezogen.

Mit Hilfe des Speicherplatzzuweisungszählers werden aufeinanderfolgenden Makrobefehlen, Konstantendefinitionen oder Speicherbereichsreservierungen lückenlos Hauptspeicherplätze zur Verfügung gestellt, d.h. die zu einem Programm gehörenden Anweisungen erhalten Speicherplätze in lückenlos aufsteigender Folge. Der Zuweisungszähler kann am Beginn des Programms bzw. in den Programmabschnitten durch die ORG-An-

weisung (s. Abschnitt 5.10.) abweichend von der beschriebenen Weise gesetzt werden.

Durch den Speicherplatzzuweisungszähler wird jeweils das linke (adreßniederwertigste) Byte des durch eine Anweisung definierten Feldes (Makrobefehl, Konstantendefinition, Speicherreservierung) adressiert.

Ein Symbol im Markenfeld einer Anweisung erhält den Zuweisungszählerstand als Adreßattribut.

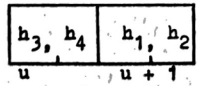
## 4.5. Zeichenketten

### 4.5.1. Zeichenketten des Typs X ($zkx_1$)

Eine Zeichenkette des Typs X wird durch ein vorangestelltes X gekennzeichnet. Die angegebene Zeichenkette wird in Apostrophe (H'27') eingeschlossen. Innerhalb der Apostrophe dürfen entsprechend i maximal 2 oder 4 Hexadezimalziffern stehen. Werden weniger angegeben, wird mit der hexadezimalen Ø aufgefüllt.

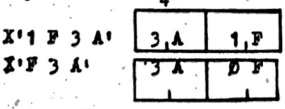
$$zkx_i \quad ::= \quad X'h\,[h\,]^{*i-1}, \quad \text{wobei } i = 2, 4$$

Die Darstellung im Speicher erfolgt, falls i = 4 in der Reihenfolge $h_3h_4$ (low-Teil), gefolgt von $h_1h_2$ (high-Teil).



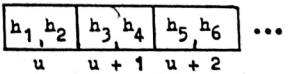

Sie dienen der CPU-gerechten Adreßdarstellung.

Beispiel für $zkx_4$:



4.5.2.
## Zeichenketten des Type H ($zkh_i$)

Eine Zeichenkette des Type H wird durch ein vorangestelltes H gekennzeichnet. Die in der Zeichenkette anzugebenden Hexadezimalziffern werden in Apostrophe eingeschlossen. Innerhalb der Apostrophe können i Hexaziffern stehen. Werden weniger als i Zeichen angegeben, so wird nach rechts mit hexadezimal Ø aufgefüllt. Das Ablegen im Speicher erfolgt in der Reihenfolge der Notierung.

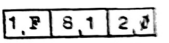



$zkh_i$ ::= H'h[h]**i-1' , wobei i = 1, 2, ...

Die H-Zeichenketten dienen der Darstellung von Zeichenketten, die nicht über Zeichenketten des Typs C angegeben werden können.

Beispiel für $zkh_6$ :

$zkh_6$ : H'1F812'



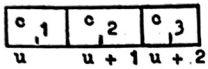

Die Byteanzahl, die von einer Zeichenkette der Länge i belegt wird, berechnet sich mit der Integer-Funktion (Int) zu

$$\text{Int}\left(\frac{i+1}{2}\right)$$

4.5.3.
## Zeichenketten des Type C ($zkc_i$)

Eine Zeichenkette des Type C wird durch ein vorangestelltes C gekennzeichnet. Die in der Zeichenkette anzugebenden KIL-Zeichen werden in Apostrophe eingeschlossen. Jedes KIL-Zeichen wird in einem Byte abgelegt. Sind weniger als i Zeichen

angegeben, so wird nach rechts mit H'ØØ' aufgefüllt. Das Ablegen der Zeichenkette in den Speicher erfolgt in der Reihenfolge der Notierung. Die Zeichenkette des Typs C dient der Darstellung von Ketten, die aus Alphazeichen bestehen.




$zkc_i$ ::= C'c [c] $*^{i+1}$, , wobei i = 1, 2, ...

Beispiel für $zkc_6$ :

C'FELDX1'

| F | E | L | D | X | 1 |
|---|---|---|---|---|---|
| u | u + 1 | u + 2 | u + 3 | u + 4 | u + 5 |

CL6'FELD'

| F | E | L | D | ØØ | ØØ |
|---|---|---|---|---|---|

## 4.6. Dezimalzahlen (c, $\bar{c}$, f, k, l, $\bar{l}$, n, nz, q, r, s)

Die mit

c, $\bar{c}$, f, k, l, $\bar{l}$, n, nz, q, r, s

angegebenen Dezimalzahlen werden nach Vorzeichenlosen (OV) und Vorzeichenbehafteten (VZ) unterschieden.

| | Vorzeichen | Wertebereich |
|---|---|---|
| c | OV | $1 \leq c \leq 16$ |
| $\bar{c}$ | OV | $1 \leq \bar{c} \leq 17$ |
| f | OV | $Ø \leq f \leq 65535$ |
| k | OV | $k \in \{4,8\}$ |
| l | OV | $1 \leq l \leq 256$ |
| $\bar{l}$ | OV | $Ø \leq \bar{l} \leq 255$ |

| | Vorzeichen | Wertebereich |
|---|---|---|
| n | OV | $\emptyset \leq n \leq 15$ |
| nz | OV | $1 \leq nz \leq 255$ |
| q | VZ | $\pm (\emptyset \leq q \leq 9999999)$ |
| r | VZ | $\pm (\emptyset \leq r \leq 999\_999\ 999\ 999\ 999)$ |
| s | VZ | $\pm (\emptyset \leq s \leq 10^{c-1}-1)$ |
| t | OV | $\emptyset \leq t < 512$ |

Die Semantik ist der jeweiligen Befehlsbeschreibung zu entnehmen.

## 4.7. Adreßdirektwert (nnnnn)

Adreßdirektwerte stellen Adreßwerte dar, die bei Übersetzung oder Programmverbindung keinen neuen Wert zugeordnet bekommen. Adreßdirektwerte werden verwendet, um absolute Adressen und Verschiebungen für die Adreßrechnung anzugeben. Damit muß der Adreßdirektwert in 2 Bytes binär verschlüsselbar sein und im Wertebereich

$$\emptyset \leq nnnnn \leq 2^{16} - 1$$

liegen.
Der Assembler akzeptiert 2 Arten von Direktwerten:

1. Dezimaler Adreßdirektwert

   $\emptyset \leq t \leq 65535$

   Der dezimale Adreßdirektwert wird als vorzeichenlose Zahl dargestellt.

2. Hexadezimaler Adreßdirektwert

   $X'\emptyset\emptyset\emptyset\emptyset' \leq zkx_4 \leq X'FFFF'$

Der maximale Wert 65535 bzw. X'FFFF' ist die größtmögliche Hauptspeicheradresse.

4.8.
## Ausdrücke (adr), Terme (term)

Jedem Operand eines Makrobefehls oder jeder Assembleranweisung, die zur Adreßdarstellung einen Ausdruck (adr) enthalten, wird bei Übersetzung ein Wert zugeordnet. Ein solcher Ausdruck kann ein einzelner Term oder eine arithmetische Verknüpfung beliebig vieler Terme sein.
Die Anzahl von Termen in einem Ausdruck wird durch die zulässige Länge der Quellzeile beschränkt.

adr ::= [±] term [± term] ...

Zusammengesetzte Ausdrücke enthalten mehrere arithmetische verknüpfte Terme. Das Klammern von Termen ist nicht zugelassen.

Treten im Operandenfeld eines Befehls mehrere Ausdrücke auf, so werden sie indexiert, z.B. adr1, adr2, adr 3.

Ein Term kann aus einem Symbol bzw. einem Adreßdirektwert oder einer Zeichenkette des Typs H mit der Länge 1, 2, 3 oder 4 bzw. einer Zeichenkette des Typs C mit der Länge 1 oder 2 oder der Angabe des aktuellen Speicherplatzzuweisungszählers bestehen.

$$\text{term} ::= \begin{Bmatrix} \text{symbol} \\ \text{nnnnn} \\ \text{zkh}_4 \\ \text{zkc}_2 \\ * \end{Bmatrix}$$

Die zur Adreßdarstellung verwendeten Ausdrücke können absolut oder verschieblich sein. Der Unterschied wird deutlich, wenn man den Wert von Adreßausdrücken bei einer Programmverschiebung durch den Programmbinder betrachtet. Der Wert eines Adreßausdrucks mit einem absoluten Ausdruck als Argument bleibt unverändert. War das Argument dagegen ein verschieblicher Ausdruck, so wird er entsprechend der Programmverschiebung modifiziert.

Ob zusammengesetzte Ausdrücke verschieblich sind oder nicht, hängt davon ab, in welcher Weise absolute und verschiebliche Terme an ihrem Aufbau beteiligt sind. Verschiebliche Terme sind Symbole (sofern nicht absolut definiert) und Bezugnahmen auf den Speicherplatzzuweisungszähler. Absolute Terme sind Adreßdirektwerte, absolut definierte Symbole und Zeichenketten vom Typ C der Länge 1 oder 2 oder H der Länge 1, 2, 3 oder 4.

### 4.8.1. Absolute und verschiebliche Ausdrücke

#### 4.8.1.1. Absolute Ausdrücke

Ein absoluter Ausdruck ist im einfachsten Fall ein absoluter Term.

Zusammengesetzte absolute Ausdrücke können absolut und auch verschiebliche Terme oder beide gemischt enthalten. Soll ein Ausdruck, der verschiebliche Terme enthält, absolut sein, so müssen folgende Regeln eingehalten werden:

1. Die verschieblichen Terme müssen in einem absoluten Ausdruck immer gepaart auftreten.
   Als gepaart gelten zwei Terme, wenn sie im gleichen Programm definiert sind und entgegengesetzte Vorzeichen tragen.

2. Gepaarte Terme brauchen im Ausdruck nicht benachbart zu stehen.

Beispiele:

```
M1-M7+118
MA+MX-MB+3
```

Die Terme MA, MB, M1, M7 seien verschieblich, der Term MX sei absolut!

Bei Symbolen, die anderen Programmen entstammen (durch EXTRN def.) hat der Assembler keine Möglichkeit, auf Paarigkeit zu prüfen. Sie sind in zusammengesetzten Ausdrücken, die ohne das externe Symbol absolut sind, möglich. Nimmt ein Term in einer Anweisung eines absoluten Ausdrucks auf den Speicherplatzzuweisungszähler Bezug, dann muß der entsprechende paarige Term dem gleichen Programm wie die Anweisung entstammen.

4.8.1.2.
Verschiebliche Ausdrücke

Ein verschieblicher Ausdruck stellt eine Hauptspeicheradresse dar, die bei einer Programmverbindung entsprechend dem vorgegebenen Ladepunkt verschoben wird.

Ein verschieblicher Ausdruck ist im einfachsten Fall ein verschiebbarer Term.
Zusammengesetzte verschiebliche Ausdrücke entstehen durch arithmetische Verknüpfung von Termen, wenn folgende Regel eingehalten wird:

> Die verschieblichen Terme müssen bis auf einen paarig auftreten. Für die Paarigkeit gelten die bei den absoluten Ausdrücken (Abschnitt 4.8.1.1.) beschriebenen Bedingungen.

Beispiel:

M1+5
M1+M7-MX
* +5
Die Terme M1, M7, MX seien verschieblich!

Symbole, die anderen Programmen entstammen (durch EXTRN definiert), sind in zusammengesetzten Ausdrücken, die ohne Betrachtung der externen Symbole einen verschieblichen Charakter aufweisen, nicht erlaubt.

Beispiel:

M1+M7-MX+EX31

M1,M7,MX sei verschiebbar, EX31 sei ein externes Symbol.

4.3.1.3.

Anwendung von Ausdrücken in absoluten und verschieblichen Programmen

In absoluten Programmen sind alle Ausdrücke absolut, d.h. sie besitzen einen festen Adreßwert im Speicher.

In verschieblichen Programmen können sowohl absolute als auch verschiebliche Ausdrücke programmiert werden.

Enthält ein Ausdruck mehrere Terme, so wird zwischen absolut und verschieblich folgendermaßen entschieden:

Die Anzahl der positiven verschieblichen Terme (PT) minus Anzahl der negativ verschieblichen Terme (NT) innerhalb eines Ausdrucks muß Ø oder 1 sein.

| | |
|---|---|
| X = PT - NT | |
| X = Ø | Absoluter Ausdruck |
| X = 1 | Verschieblicher Ausdruck |
| X ≠ Ø, 1 | Falscher Ausdruck |

Ein falscher Ausdruck wird vom Assembler MBAS 1520 (3IEX) mit einer Fehlerkennzeichnung markiert.

Bei falschen Ausdrücken wird vom Assembler kein Wert berechnet.

Beispiel:

| Term | Adreßwert<br>Wert (dezimal) |
|---|---|
| Speicherplatzzuweisungszähler | 2010 |
| I (verschieblich) | 2000 |
| J (verschieblich) | 2150 |
| XY (verschieblich) | 2200 |

| Ausdruck | Wert | |
|---|---|---|
| * | 2010 | verschieblich |
| I+5 | 2005 | verschieblich |
| -*+I | 140 | absolut |
| XY +* | 0 | falsch |
| I+*+2 | 0 | falsch |
| I-*+XY | 2050 | verschieblich |
| -I+I-XY | 0 | falsch |
| I+I-XY+* | 340 | absolut |

4.8.2.
Adreßtypen von adr

4.8.2.1.
Adreßtyp 1 (at 1)

Der Adreßtyp 1 stellt einen Ausdruck mit optionaler Längenangabe dar.

at 1 ::= $\left\{\begin{matrix}\text{adr} \\ \text{adr}\end{matrix}\right.$ $\left.\begin{matrix}\\ (1)\end{matrix}\right\}$ bzw.

in der Darstellung der Makrobefehlsbeschreibung

at 1 ::= adr [(1)]

Die explizit angegebene Länge dominiert über die implizit angegebene. Die Indirektadressierung ist erlaubt.
Im Falle, daß

```
at 1 = adr (1),
```

wird durch die Angabe von '(1)' eindeutig das Längenattribut des mit adr adressierten Bereichs vorgegeben. Für 1 gilt:

$$1 \leq 1 \leq 256$$

Bei

```
at 1 := adr
```

gelten die folgenden getroffenen Festlegungen für implizite Längen.

Explizite und implizite Längen

Befehle mit Operanden des at 1 gestatten neben der expliziten auch die implizite Längenspezifikation. In diesem Falle entfällt im Befehl die Längenangabe.

Der Assembler verwendet als Länge

bei expliziter Adressierung das Längenattribut, das eingeschlossen in "()" dem Ausdruck folgt
und bei impliziter Adressierung das Längenattribut des Ausdrucks.

Implizite Längen

Als Längenattribut eines absoluten Terms wird 1 angenommen. Das Längenattribut eines Terms, der einem Symbol im Markenfeld entspricht und einen Makrobefehl markiert, wird ebenfalls durch den Assembler zu 1 angenommen.

Bei zusammengesetzten Ausdrücken bestimmt der am weitesten links stehende Term das Längenattribut, d.h. der Ausdruck

erhält das Längenattribut des 1. Terms zugeordnet. Ist der 1. Term eine Konstante, so wird das Längenattribut gleich 1 gesetzt.

Stellt der 1. Term eine Bezugnahme auf den Speicherplatzzuweisungszähler dar, wird 1 als Längenattribut wirksam.

Bei einem über EQU definiertem Symbol wird das Längenattribut des im Operandenfeld stehenden Ausdrucks zugeordnet.

Bei Ausdrücken, die durch ein externes Symbol (nur eins ist zulässig) repräsentiert werden, erfolgt die Übernahme des Längenattributs im Bindelauf (Beachte Ausnahme bei at 3, at 4!).

Beispiele:

1. Operanden mit expliziter Längenangabe

```
                  :
                  MVC     FELDX(1Ø),FELDY
                                  ↑
                                  └── explizite Länge
                  :
FELDX:            DS      2Ø
FELDY:            DS      15
```

Es wird ab der Adresse FELDY mit der Länge l = 1Ø nach dem Feld, das mit der Adresse FELDX beginnt, transportiert.

2. Operanden mit impliziter Längenangabe

```
FELDX:            DS      1Ø
FELDY:            DS      15
FELDZ:            DS      15
                  :
A:                EQU     FELDZ
B:                EQU     FELDZ
                  :
```

```
          MVC     A,FELDY
          :
          MVC     B.FELDY
          :
```

Es wird bei beiden Befehlen ab der Adresse FELDY mit der Länge l = 15 nach dem Feld, das mit Adresse FELDZ beginnt, transportiert.

```
          PN      P 1
          :                                  Transport mit
          :                                  l = 15 ab
          MVC     FELDZ+5,FELDZ              FELDX nach
          :                                  FELDZ + 5,
          :                                  wobei FELDW
                                             mit belegt
                                             wird
          MVC     5+FELDZ,FELDZ              Transport mit
          :                                  l =  1 ab
          :                                  FELDX
FELDX:    DS      10
FELDZ:    DS      15
FELDW:    DS       5

          PN      P2
          :
          MVC     FELDX,FELDY
          :
FELDX:    DS      2Ø
FELDY:    DS      15
```

Den Operanden FELDX und FELDY des MVC-Befehls wird das Längenattribut (LA = 2Ø) der Bereichsdefinition für FELDX und FELDY zugewiesen. Damit wird mit der Länge 2Ø Byte (implizite Länge) der MVC-Befehl bei Abarbeitung ausgeführt werden.

4.8.2.2.
Adreßtyp 2 (at 2)

Der Adreßtyp 2 läßt nur Ausdrücke zu.
Längenangaben sind nicht erlaubt.

at 2 ::= adr

Indirektadressierung ist erlaubt.

4.8.2.3.
Adreßtyp 3 (at 3)

Der Adreßtyp 3 läßt Ausdrücke mit optionaler Längenangabe für gepackte BCD-Daten zu.

at 3 ::= $\left\{\begin{matrix}\text{adr (k)}\\ \text{adr}\end{matrix}\right\}$ bzw. in der

Darstellung der Makrobefehlsbeschreibung

at 3 ::= adr [(k)]

Die explizite Längenangabe dominiert über die implizite (vgl. Abschnitt 4.8.2.1.).

Bei Gebrauch externer Symbole ist die Länge immer explizit zu spezifizieren. Die Längenangabe k kann nur die Werte 4 oder 8 annehmen (k $\in \{4, 8\}$ ).

Indirektadressierung ist erlaubt.

4.8.2.4.
Adreßtyp 4 (at 4)

Der Adreßtyp läßt einen Ausdruck mit eingeschränkter Längenangabe zu. Die Längenangabe ist optional. Damit dominiert die explizit angegebene Länge über die implizite.

at 4 ::= $\left\{\begin{array}{l}\text{adr (l)}\\ \text{adr}\end{array}\right\}$ , wobei $l \leq 64$

bzw. in der Darstellung der Makrotefehlsteschreibung

at 4 ::= adr [(l)] , wobei $l \leq 64$.

Bei Gebrauch externer Symbole ist die Länge explizit zu spezifizieren.
Indirektadressierung ist erlaubt.

4.8.2.5.
Adreßtyp 5

Der Adreßtyp 5 läßt nur Ausdrücke ohne Längenangabe und ohne Indirektadressierung zu.

4.8.3.
Indirekte Adressierung

Durch Einklammern der Adreßtypen at 1, at 2, at 3 und at 4 in runde Klammern "()" ist in allen zutreffenden Befehlen eine Indirektaddressierung möglich.

(ausdruck)
(ausdruck (q))
q ::= l/k

Die expliziten Längenangaben l und k sind in die Indirektklammern aufzunehmen.

## 4.9. Direktoperanden

Ein Direktoperand ist eine Konstante, die vom Befehl verarbeitet wird. Sie wird im Befehl angegeben.

### 4.9.1. Direktoperand der Länge 1 Byte (d 1)

Ein Direktoperand der Länge 1 Byte kann sein:

d 1 ::= at 5,

wobei der Wert des Adreßattributs $\leq 255$ und absolut sein muß.

### 4.9.2. Direktoperand der Länge 2 Byte (d 2)

Ein Direktoperand der Länge 2 Byte kann sein:

d 2 ::= at 5,

wobei der Wert des Adreßattributs absolut sein muß.

## 4.10. Absoluter Ausdruck mit eingeschränktem Wert (g)

Es gilt:

g ::= adr,

wobei der Wert des Ausdrucks absolut und $\leq 255$ sein muß, d.h. Adreßattribut $\leq 255$.

4.11.

Sprungbedingung in Abhängigkeit von einem der Selektoren 0 - 15 (s 15)

s 15 ::= $\left\{\begin{matrix} n \\ Nn \end{matrix}\right\}$ , wobei

n Selektornummer

N Negation, Umkehrung der Sprungbedingung

Beispiel:

| s 15 | Bedeutung/Bedingung |
|---|---|
| 7 | "Wenn Selektor 7 gesetzt" |
| N15 | "Wenn Selektor 15 nicht gesetzt" |

4.12.

Spezifikation der Bitpositionen des K-, S- und KR-Registers (l 15)

l 15 ::= $\left\{\begin{matrix} n \\ n, \text{ l } 15 \end{matrix}\right\}$ , wobei

n die Bitposition im entsprechenden Register beschreibt (Ø für niederwertigstes Bit, 15 für höchstwertiges Bit). Nach der Definition von l 15 ist eine Angabe als Liste der Bitpositionen möglich. Die Trennung der Bitpositionen erfolgt durch Komma ",".

Beispiel:

Ø , 3 , 5 , 7

Die als Liste angegebenen Bitpositionen gelten im Sinne einer logischen Konjunktion. Anhand des Beispiels ergibt sich

Bits: Ø und 3 und 5 und 7

## 4.13. ON-Codes (onc)

Der ON-Code spezifiziert die Fehlerbedingung für den ON-Befehl (Verzweigung zur spezifizierten Adresse, wenn angemeldete Fehlerbedingung eingetreten ist).

$$onc ::= \left\{ \begin{matrix} E \\ D \\ Z \\ P \end{matrix} \right\}$$

- E Systemfehler mit Abbruchwirkung
- D Datenfehler bei BCD-Arithmetik
- Z Nulldivision
- P Durchsuchen des Staks nach der Meldung "Netzausfall" (nur für Schalterterminal K 8924)

## 4.14. ONF-Codes (onfc)

Der ONF-Code spezifiziert die Bedingung, unter der die durch einen ONF-Befehl angemeldete Verbindung zwischen einer Datei und einem Unterbrechungsprogramm wirksam wird.

$$onfc ::= \left\{ \begin{matrix} ERR \\ EOF \\ EOV \\ MRK \\ DT \end{matrix} \right\}$$

| onfo | Bedeutung |
|---|---|
| ERR | Fehler (error) |
| EOF | Dateiende (end of file) |
| EOV | Datenträgerende (end of volume) |
| MRK | Markierter Satz (marked) |
| DT | Datenfernübertragung (data transmission) |

4.15.
Parameter für E/A-Operationsspezifikation (par/W)

par ::= $\left\{\begin{matrix} W \\ M \\ WM \end{matrix}\right\}$

Die Angabe von par ist optional, d.h. sie kann entfallen. Je nach Datei-E/A-Befehl richtet sich die Zulässigkeit des Parameters M.

Ein

W

als Operand einer E/A-Operation ohne Dateifunktion besitzt die gleiche Bedeutung wie par ::= W.

Die W-Angabe ist optional.

| par | Bedeutung |
|---|---|
| W | Kein Warten auf Ausführung des Befehls (WAIT- bzw. WAITC-Befehl folgt im Programm, d.h. Simultanarbeit E/A-Prozeß zum Programm) |
| M | Schreiben der Marke "Markierter Satz" (nur bei WRITE, WRITD zugelassen) beim Schreiben von Daten |
| WM | W und M werden gleichzeitig angewiesen! |
| keine Angabe | Erfolgt keine Angabe der Parameter par, wird vom Assembler der E/A-Befehl, ohne Simultanarbeit, d.h. mit Warten auf Ausführung, generiert. |

## 4.16. Positionierung bei sequentiellen Datei-E/A-Befehlen (pos)

pos dient der Positionierung von logischen Sätzen innerhalb eines sequentiell organisierten Datenbestandes.

pos ::= P / N / B / L / E

| pos | Bedeutung | |
|---|---|---|
| P | first: | Positionierung auf 1. Satz des Datenbestands |
| N | next: | Positionierung auf nächsten ((n + 1)-ten) Satz des Datenbestands |
| B | back: | Positionierung auf den gerade bearbeiteten Satz des Datenbestands |
| L | last: | Positionierung auf den letzten aufgezeichneten Satz im Datenbestand (bei FD) |
| E | extended: | Positionierung auf den nächsten freien Satz im Datenbestand |

## 4.17. Positionierparameterliste (pparlist)

Die Angabe von pparlist erfolgt beim POS-Befehl (z.B. für Druckkopf- oder Kursorpositionierung), wobei gilt:

pparlist ::= pparlist, ppar/ppar

$$\text{ppar} ::= \left\{ \begin{array}{l} H \left\{ \begin{array}{c} + \\ - \\ = \end{array} \right\} g \\ \left\{ \begin{array}{l} V \left\{ \begin{array}{c} + \\ - \\ = \end{array} \right\} g \\ VT \end{array} \right\} \\ I = adr \\ FF \\ R \end{array} \right\}$$

Koordinatenpositionierung:

| | Bedeutung |
|---|---|
| H | Horizontal |
| V | Vertikal |
| + | Relativ vorwärts |
| - | Relativ rückwärts |
| = | Absolut |
| g | Absoluter Ausdruck mit eingeschränktem Wert ($\leq 255$), s. Abschnitt 4.9. |

VT - Zeilensteuerfunktion
Vertikaltabulation bis Stoppunkt entsprechend der durch den Befehl CD definierten Steuertabelle, ansonsten absolute Vertikalpositionierung

FF - Neue Seite
In Abhängigkeit vom Gerät erfolgt die Ausführung. (s. POS-Befehl)

R - Anweisen Umkehrfunktion
Diese Funktion wird gerätespezifisch ausgeführt. (s. POS-Befehl)

I = adr - Positionierung erfolgt in Verbindung mit den Werten einer 2-Byte-Zelle, die durch einen Ausdruck adr adressiert wird.

**Beispiel:**

```
                .
                .
                .
                POS Ø,H-Ø, I=POSPA
                .
                .
POSPA:          DC  I'5'   ; H - Zeile
                DC  I'2'   ; V - Zeile
```

Es werden 5 Horizontalschritte rückwärts und 2 Vertikalschritte vorwärts auf dem logischen Gerät "Ø" ausgeführt.

**Beachte:**

Wird H bzw. V nicht in pparlist spezifiziert, so wird vom Assembler automatisch

$$H + Ø \quad \text{bzw.} \quad V + Ø$$

generiert.
Verboten ist die Angabe

$$\text{pparlist:} = \left( \begin{matrix} H \left\{ \begin{matrix} + \\ - \\ = \end{matrix} \right\} g \\ V \left\{ \begin{matrix} + \\ - \\ = \end{matrix} \right\} g \end{matrix} \right), \quad I = adr$$

mit $g \neq Ø$.
Der Assembler bringt eine Fehlermarkierung.

Weitere Besonderheiten sind der Befehlsbeschreibung des POS-Befehls zu entnehmen.

## 4.18. Funktionscodes (fctc)

Die nachfolgenden Funktionscodes sind nur in Verbindung mit dem CTRL-Befehl zugelassen und lösen Hardwarefunktionen aus (s. Beschreibung CTRL-Befehl).

In der Sprache sind für foto zugelassen:

foto: := RBC/WTM/FOR/REW/BSR/BBC/BSF/FSF/TRP/CUT/
FLP/BLP/KPN/KPI/SDC/BDC /FL2/BL2/BSS/DEL/
WON/WOF/RND/NND/PUO/BWR/SPO/CFC/STS/GBO

## 4.19. Starttastencodes (stc)

Mit Hilfe von stc lassen sich die Starttastencodes über die Tastenbezeichnung angeben. Eine Zulässigkeitsprüfung für einen speziellen Gerätetyp wird nicht vorgenommen. Beachte deshalb die Übersicht zu den Starttastencodes unter den Befehlen BCA/BCN.

stc ::= ST/0/1/2/3/4/.../63/
ET1/ET2/ET/SEND/SKIP/REC/
PA1/PA2/PA3/CLEAR/CNCL/ENTER/CU/CA/FV/FB

Anmerkung: FV ≙ F; FB ≙ F

## 4.20. Maske (m)

Bei den Befehlen PUTP, UNPK ist über den Operanden m eine Maske zu definieren.

m ::= [int] [deza] fs ds

- int Interpunktionsangabe
- deza Angabe der Dezimalstellenabtrennung
- fs Füllnullenbehandlung
- ds Angabe der Zahl der abzutrennenden Dezimalstellen

Interpunktionsangabe:

int ::= NI/HP/HS/TP/TS

wobei

| | |
|---|---|
| NI | Keine Interpunktion |
| HP | Hunderter-Punkt |
| HS | Hunderter-Space |
| TP | Tausender-Punkt |
| TS | Tausender-Space |

Angabe der Dezimalstellenabtrennung:

deza ::= C/P

wobei

| | |
|---|---|
| C | Komma |
| P | Punkt |

Füllnullenbehandlung:

fs ::= Z/S/*

| | |
|---|---|
| Z | Füllnullen werden als Ø ausgegeben |
| S | Füllnullen werden als Space ausgegeben |
| * | Füllnullen werden als * ausgegeben |

Vornullen werden stets als "Ø" ausgegeben.
Angabe der Zahl abzutrennender Dezimalstellen

$0 \leq ds \leq 3$

Zusammenfassung:

m ::= [NI / HP / HS / TP / TS] [C / P] [Z / S / *] [0 / 1 / 2 / 3]

4.21.

Konstantenwert (wert)

wert ::= { F'f' | W'q' | D'r' | CLl'$c_1 \ldots c_l$' | { zkc$_1$ | C'$c_1$ ... $c_l$' } | { zkh$_t$ | H'$h_1$ ... $h_t$' } | NLC̄'s' | N'r' | zkx$_4$ | { X'$h_1h_2h_3h_4$' | I'Ī' } | V'symbol' | A'adr' }

wert tritt nur im Operandenfeld von Konstantenanweisungen auf und dient der Definition einer Konstanten (s. Abschnitt 5.8.).

## 5.
## Beschreibung der MABS 1520 - Pseudobefehle

Makrobefehle vertreten im symbolischen Programm die Befehle, die zum Programmausführungszeitpunkt die Arbeit des Geräts steuern.

Die Pseudobefehle sind nur während der Übersetzung des Quellprogramms wirksam. Jedoch liefern die meisten, z.B. Pseudobefehle, die zur Konstantendefinition und Speicherplatzreservierung im Quellprogramm genutzt werden, einen echten Beitrag zum Aufbau des Objekt- oder Phasenprogramms.

### 5.1.
### Beginn eines Quellprogramms

Format:

| Markenfeld | OP-Codefeld | Operandenfeld |
|---|---|---|
| Keine Eintragung | PN | symbol |

PN darf, wenn vorhanden, nur am Programmanfang stehen und kennzeichnet den Programmbeginn. Der Pseudobefehl PN muß nicht unbedingt angegeben werden. Der 1. Befehl eines Programms ist auch sein Startpunkt.

Hinweise:
Wird das Programm mit einem Makrobefehl begonnen, muß diesem Befehl ein SOP vorangehen.

Wird ein Programm mit einem Vereinbarungsteil begonnen (DS- und DC-Vereinbarungen), muß dieser Bereich mit einem Sprung übergangen werden.

Beispiel:

```
         PN      PROGA
         SOP
         B       MARKE
FELL 1:  DS      2Ø                  , BEGINN VEREINBARUNGSTEIL
KONS:    DC      I'1ØØ'
                 :
                 :                   , ENDE VEREINBARUNGSTEIL
MARKE:   X       FELD1,FELD1
                 :
                 :
```

## 5.2. Ende eines Quellprogramms

Format:

| Markenfeld | OP-Codefeld | Operandenfeld |
|---|---|---|
| Keine Eintragung | END | Keine Eintragung |

Die END-Anweisung kennzeichnet, wenn vorhanden, das Programmende. Sie muß nicht unbedingt angegeben werden.

## 5.3. Seitenschaltung im Übersetzungsprotokoll

Format:

| Markenfeld | OP-Codefeld | Operandenfeld |
|---|---|---|
| Keine Eintragung | EJEC | Keine Eintragung |

Beim Erkennen einer EJEC-Anweisung durch den Assembler wird das Assemblerprotokoll mit einer neuen Seite fortgesetzt.

## 5.4. Zeilenvorschub im Übersetzungsprotokoll

Format:

| Markenfeld | OP-Codefeld | Operandenfeld |
|---|---|---|
| Keine Eintragung | SPACE | t |

Die im Operanden der Anweisung anzugebende Dezimalzahl t spezifiziert die Anzahl der Leerzeilen, die in das Assemblerprotokoll eingefügt werden.

$0 \leq t \leq 255$

Wenn t > als Leerzeilen bis zum spezifizierten Blattende bleiben, wirkt SPACE t wie EJEC.

## 5.5. Kennzeichnung externer Symbole

Format:

| Markenfeld | OP-Codefeld | Operandenfeld |
|---|---|---|
| Keine Eintragung | EXTRN | symbollist |

Die EXTRN-Anweisung kennzeichnet Symbole, die in diesem Quellmodul benutzt werden, aber in einem anderen definiert wurden (Verbindungssymbole).

symbollist := symbollist,symbol/symbol

Die Symbolliste symbollist enthält ein oder mehrere Symbole, getrennt durch Kommata. Die Kapazität der Symbolliste wird durch die Quellzeilenlänge beschränkt.

Die Symbole im Operandenfeld dürfen im gleichen Quellmodul nicht im Markenfeld einer Befehlszeile erscheinen.
Das Markenfeld darf keine Eintragung enthalten.

Beachte:
Symbole aus V-Typ-Adreßkonstanten (s. Abschnitt 5.8.) müssen nicht durch EXTRN-Anweisungen gekennzeichnet werden.

Die Herstellung der Verbindung zwischen Quellmoduln ist unter Abschnitt 5.6. erläutert.

## 5.6. Definition symbolischer Eintrittspunkte

Format:

| Markenfeld | OP-Codefeld | Operandenfeld |
|---|---|---|
| Keine Eintragung | ENTRY | symbollist |

Eine ENTRY-Anweisung kennzeichnet Symbole als Verbindungssymbole, die in einem Quellmodul definiert, aber von einem anderen (getrennt von ihm übersetzten) angesprochen werden sollen.

Das sind Symbole, die die Verbindung von getrennt übersetzten Quellmoduln ermöglichen sollen.

symbollist ::= symbollist, symbol/symbol

Die Symbolliste symbollist enthält ein oder mehrere Symbole, getrennt durch Kommata. Die Kapazität der Symbolliste wird durch die Quellzeilenlänge beschränkt.
Das Markenfeld darf keine Eintragung enthalten.
Es sind mehrere ENTRY-Anweisungen erlaubt.
Die Summe der Symbole, die in einem Quellmodul durch EXTRN- und ENTRY-Anweisungen sowie Definition von V-Adreßkonstanten vereinbart werden, darf 256 nicht überschreiten.
Alle genannten Symbole müssen im Programm definiert sein.

## Symbolische Verbindung zwischen Quellmoduln

Für Quellmoduln, die getrennt übersetzt und erst danach durch den Binder (LINK 1520(SIEX)) zu einem Programm verbunden werden sollen, besteht das Problem, gewünschte Verbindungen symbolisch zu definieren. Das wird durch die Verbindungssymbole erreicht.

Zu deren Definition und Kennzeichnung dienen die in diesem Abschnitt und in Abschnitt 5.5. beschriebenen Anweisungen.

Der Assembler stellt dem Binder Informationen über die Verbindungssymbole zur Verfügung. Soll in einem Quellmodul ein im Markenfeld eines Befehls bzw. einer Anweisung auftretendes Symbol zugleich als Verbindungssymbol definiert werden, muß das dem Assembler durch eine ENTRY-Anweisung mitgeteilt werden. Diese Anweisung spezifiziert den Eintrittspunkt, der von einem anderen Quellmodul zur Bezugnahme auf einen Befehl oder auf Daten verwendet wird.

Entsprechend muß dazu ein Quellmodul, der ein Symbol eines anderen Quellmoduls verwendet, dieses als extern über eine EXTRN-Anweisung kennzeichnen.

Der Assembler ordnet einem externen Symbol ein Adreßattribut gleich Ø und ein Längenattribut gleich Ø zu und trägt dies in das Verzeichnis der externen Symbole ein.

Die symbolische Verbindung kann ebenfalls durch eine V-Adreßkonstante realisiert werden. Die V-Adreßkonstante wird bei externen Verzweigungen verwendet und nicht für den Zugriff auf Daten (Näheres s. Abschnitt 5.8.).

**Beispiel:**

**1. Quellmodul**

```
                PN              A
                :
MARK1:          A               FELD1,FELD2
                :
                M               2,FELD1,KONX
                :
MARK3:          PUT             Ø,FELD1(16),Ø,W
                :
                B               MARK2
                :
FELD1:          DS              6
                ENTRY           MARK3, MARK1,
                EXTRN           FELD2,KONX,MARK2
                :
                END
```

**2. Quellmodul**

```
                PN              B
MARK2:          ENT             FELD2(6)
                :
                C               FELD2,KONX
                BP              MARK1
                :
                B               MARK3
                :
KONX:           DC              F'1Ø76'
FELD2:          DS              6
                ENTRY           MARK2,FELD2,KONX
                EXTRN           MARK3,MARK1
                :
                END
```

## 5.7. Speicherbereichsdefinition

Format:

| Markenfeld | OP-Codefeld | Operandenfeld |
|---|---|---|
| [symbol:] | DS | adr |

Erweiterte DS-Anweisung:

| Markenfeld | OP-Codefeld | Operandenfeld |
|---|---|---|
| [symbol:] | DS | adr*adr<br>Produkt der 2 Werte = z; wenn 1. Wert fehlt, so wird implizit = 1 angenommen. |

Die im Operandenfeld auftretenden Adreßausdrücke adr müssen vom Adreßtyp at 5 sein und einen absoluten Wert darstellen. Die DS-Anweisung dient zum Reservieren und eventuell zum Benennen von Speicherbereichen. In adr gebrauchte Symbole müssen vorher spezifiziert worden sein.

z gibt die Anzahl der frei zu haltenden Bytes an.

$1 \leq z \leq 65535$

Der Assembler erzeugt für den frei zu haltenden Bereich keinen Inhalt, es wird nur der Speicherplatzzuweisungszähler um den Wert von z erhöht.

Beim Laden des Programms aus der Phasenbibliothek bleibt der (zufällige) Hauptspeicherinhalt (RAM) in diesem Bereich erhalten. Sinnvolle Daten kommen im allgemeinen erst während der Programmausführung auf die reservierten Hauptspeicherplätze. Bereiche, die einen definierten Anfangswert enthalten sollen, müssen mit der DC-Anweisung definiert werden bzw. sind bei Programmabarbeitung zu erzeugen.

Ein Symbol im Markenfeld erhält das Adreßattribut gleich dem Wert des Speicherplatzzuweisungszählers und das Längenattribut gleich dem Wert des 2. Ausdrucks. Falls dieser Wert > 256, so erhält das Symbol das Längenattribut = 1. Bei Gebrauch des Symbols im Operanden dominiert die explizit angegebene Länge.

Beispiel:

```
FELD:  DS      4Ø      ; RESERVIERUNG 4Ø BYTE UNTER SYMB.
                       ; NAMEN 'FELD'
```

5.8.
Konstantendefinition

Format:

| Markenfeld | OP-Codefeld | Operandenfeld |
|---|---|---|
| [symbol:] | DC | wert |

Die DC-Anweisung dient der Definition von Konstanten im Programm. Der Assembler unterscheidet 13 Typen von Konstanten.

Die im Operandenfeld angegebenen Konstanten werden im Programm abgelegt und sind damit für eine spätere Programmabarbeitung bereitgestellt.

Wird ein Symbol im Markenfeld der Anweisung verwendet, so besteht die Möglichkeit, in den Operanden anderer Anweisungen (Makrobefehle und Pseudobefehl EQU) symbolisch auf diese Konstante Bezug zu nehmen.

Ein im Markenfeld stehendes Symbol erhält ein Adreßattribut und ein Längenattribut zugeordnet.

Der Typ der Konstante wird zuerst im Operanden angegeben. Es folgt, in Apostrophe eingeschlossen, die Konstante.

Leerzeichen zwischen Typangabe und erstem Apostroph sind nicht zugelassen. Mit einer Anweisung kann nur eine Konstante definiert werden.

Festkommakonstante (Typ F)

Format:

| Markenfeld | OP-Codefeld | Operandenfeld |
|---|---|---|
| [symbol] | DC | F'f' |

$0 \leq f \leq 65535$
$(0 \leq f \leq 2^{16}-1)$

Die in der Anweisung definierte Konstante wird in eine Binärzahl der Länge 2 Byte überführt und in der Reihenfolge low-Teil, high-Teil bereitgestellt.

| low-Teil | high-Teil |
|---|---|
| zz | zz + 1 |

zz - aktueller Wert des Speicherplatzzuweisungszählers

Ein im Markenfeld angegebenes Symbol erhält das Längenattribut 2.

BCD-Konstante der Länge 4 (Typ W)

Format:

| Markenfeld | OP-Codefeld | Operandenfeld |
|---|---|---|
| [symbol] | DC | W'q' |

Die Anweisung definiert eine BCD-Konstante (gepacktes Format) der Länge 4 Byte.

$\pm$ $(0 \leq q \leq 10^7-1)$

Wird kein Vorzeichen angegeben, so wird automatische "+" angenommen.

(Beispiel: DC W'-137')

Ein im Markenfeld angegebenes Symbol erhält das Längenattribut 4

BCD-Konstante der Länge 8 Byte (Typ D)

Format:

| Markenfeld | OP-Codefeld | Operandenfeld |
|---|---|---|
| [symbol:] | DC | D'r' |

Die Anweisung definiert eine BCD-Konstante (gepacktes Format) der Länge 8 Byte.

$\pm (0 \leq r \leq 10^{15} - 1)$

Wird kein Vorzeichen angegeben, so wird automatisch "+" angenommen.

Beispiel: DC D'+15873008169'

Ein im Markenfeld angegebenes Symbol erhält die Länge 8.

Zeichenketten einer vereinbraten Länge 1 (Typ CL)

Format:

| Markenfeld | OP-Codefeld | Parameterfeld |
|---|---|---|
| symbol: | DC | CL1'$c_1$ ... $c_1$' |

Die Anweisung definiert eine Zeichenkette aus KOI-7-Bit-Code-Zeichen mit einer vereinbraten Länge von 1 Bytes, wobei

$1 \leq l \leq 256$

(bzgl l vlg. Abschnitt 4.6.)

Werden weniger als l vereinbarte Zeichen angegeben, so wird die Konstante linksbündig eingetragen. Rechts wird mit Füllnullen (NUL:=H'ØØ') aufgefüllt.

**Beachte:**

Durch den Editor wird die Anzahl der in der Zeichenkette angebbaren Zeichen entsprechend der Quellzeilenlänge beschränkt.

Mit CL l'' können Speicherbereiche mit der definierten Länge l und dem Byteinhalt H'ØØ' (gelöschte Bereiche) erzeugt werden. Werden mehr als l vereinbarte Zeichen angegeben, so wird die Konstante linksbündig eingetragen. Die überzähligen Zeichen fallen weg.

Beachte:

Besonderheit für das Zeichen ('). Dieses Zeichen ist in der Anweisung jeweils zweimal zu schreiben, wenn es in der internen Form jeweils einmal erscheinen soll.

Darstellung von nur Apostroph mit "CL 1''''".

Ein im Markenfeld angegebenes Symbol erhält als Längenattribut die vereinbarte Länge l zugewiesen.

Beispiel:

| | | |
|---|---|---|
| MARKE: | DC | CL 16'VEB ROBOTRON BWK' |

Zeichenkette ohne Längenvereinbarung (Typ C)

Format:

| Markenfeld | OP-Codefeld | Operandenfeld |
|---|---|---|
| [symbol:] | DC | $C'c_1 \ldots c_l'$ |

Die Anweisung definiert eine Zeichenkette aus KOI-7-Bit-Code-Zeichen, d.h. sie wird wie angegeben abgelegt.

$1 \leq l \leq 256$

Bei Verwendung des Apostroph ist eine Besonderheit zu beachten (s. Konstante Typ CL). Die Darstellung von nur Apostroph erfolgt mit "C''''".

Ein Symbol im Markenfeld der Anweisung erhält das Längenattribut l.

Beachte:

Durch den Editor wird die Anzahl der in der Zeichenkette angebbaren Zeichen entsprechend der Quellzeilenlänge beschränkt.

## Hexadezimalkonstante (Typ H)

**Format:**

| Markenfeld | OP-Codefeld | Operandenfeld |
|---|---|---|
| [symbol:] | DC | H'$h_1$ ... $h_i$' |

Die Anweisung definiert eine Zeichenkette aus Hexadezimalziffern der Länge i, die in

$$\text{Int}\left(\frac{i+1}{2}\right) \quad \text{Bytes}$$

abgelegt sind, wobei

$$i = t$$

mit

$$0 \leq t \leq 512.$$

**Beachte:**

Durch den Editor wird die Anzahl der in der Zeichenkette angebbaren Zeichen entsprechend der Quellzeilenlänge beschränkt.

Ein Symbol im Markenfeld erhält das Längenattribut t.

## Ungepackte Dezimalzahl mit vereinbarter Länge (Typ NL)

**Format:**

| Markenfeld | OP-Codefeld | Operandenfeld |
|---|---|---|
| [symbol:] | DC | NL $\bar{c}$'s' |

Die Anweisung definiert eine Dezimalkonstante mit vereinbarter Länge im ungepackten Datenformat.

$$\pm\,(0 \leq s \leq 10^{\bar{c}-1} - 1)$$

Die Stellenzahl der Dezimalkonstanten ( $\bar{c}$) ist unmittelbar nach der Typangabe im Operanden zu vereinbaren.

Leerzeichen zwischen Typangabe, Stellenzahl $\bar{c}$ und Konstante 's' sind nicht zulässig.

$$1 \leq \bar{c} \leq 17$$

In die Stellenzahl geht das Vorzeichen ein. (Bsp.: DC NL 4' - 128').

Bei Angabe von weniger als $\bar{c}$ vereinbarten Stellen werden die Reststellen nach links mit NUL (H'ØØ') aufgefüllt.
Erfolgt keine Vorzeichenangabe, wird automatisch "+" angenommen, d.h. ein Byte wird immer für das Vorzeichen verwendet.
Ein Symbol im Markenfeld erhält das Längenattribut $\bar{c}$.

Ungepackte Dezimalzahlen ohne Längenvereinbarung (Typ N)

Format:

| Markenfeld | OP-Codefeld | Operandenfeld |
|---|---|---|
| [symbol:] | DC | N'r' |

Die Anweisung definiert eine Dezimalkonstante im ungepackten Datenformat.

$$\pm\ (0 \leq r \leq 10^{15} - 1)$$

Die Länge der Konstanten resultiert aus der Zahl der spezifizierten Ziffern plus Vorzeichen. Wird kein Vorzeichen angegeben, wird automatisch "+" angenommen.
(Bsp.: DC N' + 2Ø51Ø7')

Ein Symbol im Markenfeld erhält ein Längenattribut entsprechend der Zeichenanzahl.

## A - Adreßkonstante (Typ A)

Format:

| Markenfeld | OP-Codefeld | Operandenfeld |
|---|---|---|
| [symbol:] | DC | A(adr) |

Die Anweisung definiert eine Adreßkonstante über einen absoluten oder verschiebbaren Ausdruck.

Die Adreßkonstante wird in 2 Byte abgelegt.
Wird eine Adreßkonstante über einen verschieblichen Ausdruck definiert, so erhält sie ihren endgültigen Wert bei Programmverbindung.

Für die A-Adreßkonstante wird das Längenattribut 1 angenommen.

## Y-Adreßkonstante (Typ Y)

Format:

| Markenfeld | OP-Codefeld | Operandenfeld |
|---|---|---|
| [symbol:] | DC | Y(adr) |

Diese Anweisung definiert eine Konstante, die den L-Teil des Adreßwertes eines absoluten oder verschieblichen Ausdruck angibt. Die Adreßkonstante wird in 1 Byte abgelegt.

Wird eine Adreßkonstante über einen verschieblichen Ausdruck definiert, so erhält sie ihren endgültigen Wert bei Programmverbindung.

Für die Y-Adreßkonstante wird das Längenattribut 1 angenommen.

Adreßdirektwert der Länge 2 Byte (Typ X)

Format:

| Markenfeld | OP-Codefeld | Operandenfeld |
|---|---|---|
| [symtol:] | DC | $zxx_4$ |

Durch die Anweisung wird ein Adreßdirektwert definiert. Ein Symbol im Markenfeld erhält das Längenattribut 2.

Integerkonstante (Typ I)

Format:

| Markenfeld | OP-Codefeld | Operandenfeld |
|---|---|---|
| [symbol] | DC | I'1' |

Es wird eine Konstante mit

$$0 \leq 1 \leq 255$$

definiert, die in einem Byte abgelegt wird. (Esp.: DC I'14').

Ein Symbol im Markenfeld erhält das Längenattribut 1.

V-Adreßkonstante (Typ V)

Format:

| Markenfeld | OP-Codefeld | Operandenfeld |
|---|---|---|
| [symbol] | DC | V(symbol) |

Die V-Adreßkonstante wird verwendet, um Adreßattribute von verschieblichen Symbolen aus anderen Quellmoduln (die nicht im selben Übersetzungslauf verarbeitet werden) greifbar zu machen. Das Symbol in der V-Adreßkonstante muß nicht zusätz-

lich in der EXTRN-Anweisung benannt werden. Die Symbole müssen im externen Modul durch ENTRY definiert sein. (Symbole in DC-V-Anweisungen werden immer als extern angesehen.)

Die Symbole sollten externe Sprungziele benennen, denn zur Bezugnahme auf externe Datenfelder sind sie nicht geeignet. Ein Symbol im Markenfeld erhält das Längenattribut 2.

Beispiel:

| | |
|---|---|
| DC | V'MARK2' |

## 5.9. Wertzuweisung für ein Symbol

Format:

| Markenfeld | OP-Codefeld | Operandenfeld |
|---|---|---|
| symbol: | EQU | adr |

Durch die EQU-Anweisung werden einem Symbol, das im Markenfeld der Anweisung steht, Adreß- und Längenattribut des im Operandenfeld angegebenen Ausdrucks zugeordnet. Das Symbol übernimmt den Charakter - absolut oder verschieblich - des Ausdrucks im Operanden.

Enthält der Ausdruck Symbole, so müssen diese alle in vorhergehenden Anweisungen des gleichen Übersetzungslaufes definiert worden sein.

Stellt der 1. Term eine Bezugnahme auf den Zuweisungszähler oder einen Direktwert dar, dann wird das Längenattribut 1 angenommen.

Externe Symbole sind nicht zugelassen.
Als Adreßattribut erhält das Symbol den Adreßwert des Operanden. Das Längenattribut wird vom 1. Term des Operanden übernommen. Die EQU-Anweisung ermöglicht es, zusammengesetzte Ausdrücke oder Direktwerte zu benennen.

**Beispiel:**

```
L1:     EQU     *-START+TAB
KON:    EQU     258
A:      EQU     B
```

## 5.10. Setzen des Speicherplatzzuweisungszählers

**Format:**

| Markenfeld | OP-Codefeld | Operandenfeld |
|---|---|---|
| Keine Eintragung | ORG | adr |

Der Adreßausdruck adr muß vom Adreßtyp at 5 und absolut sein. Alle in adr gebrauchten Symbole müssen vorher definiert worden sein.

Die ORG-Anweisung bewirkt das Einstellen des Zuweisungszählers auf einen gewünschten Wert. Ab ORG-Anweisung wird das Programm absolut übersetzt, d.h. das Programm gilt als absolut, falls ORG als 1. Befehl steht.

## 5.11. Definition des statischen Dateidefinitionsblocks (DDB)

**Format:**

| Markenfeld | OP-Codefeld | Operandenfeld |
|---|---|---|
| symbol: | DDB | ddbparlist |

Die DDB-Anweisung bewirkt den Aufbau eines Dateidefinitionsblocks (statischer Dateidefinitionsblock).

```
ddbparlist ::=  ddbparlist,ddbpar/ddbpar
ddbpar     ::=  schlüsselwort = spezifikation
```

Die Reihenfolge der ddbpar ist beliebig.
Nicht spezifizierte Parameter (ddbpar) der ddbparlist werden standardmäßig (s. Spalte "Standard" der DDB-Parameter-Tabelle) generiert.

Parameter dürfen nicht getrennt werden, d.h. sie müssen komplett auf einer Quellzeile stehen und werden, falls die Quellzeile für die Parameterliste nicht ausreicht, in Folgezeilen eingetragen. Ein Komma nach dem letzten Parameter jeder Zeile ist nicht erforderlich. Die Folgezeile ist mit '*' zu kennzeichnen.

DDB-Parameter-Tabelle

Schlüsselwort

| DCB | Zeiger auf den dynamischen Dateidefinitionsblock | |
|---|---|---|
| | DCB = adr<br><br>Keine Indirektadressierung erlaubt | Standard:<br>Automatische Generierung von DCB einschließlich der Ausführung der DCB-Pseudoanweisung |
| NA | Physischer Dateiname<br>NA = { symbol 8 / (adr,l [,l] * 4) / zkh$_8$ }<br>symbol 8::= symbol bestehend aus maximal 8 Zeichen | Standard:<br>H'ØØØØØØØØØØØØØØØØ' |
| DT | Dateityp [1]<br>DT = dt<br>dt ::= { R / RW }<br>R Lese-Datei<br>RW Lese-Schreib-Datei | Standard:<br>R |

| | | |
|---|---|---|
| | SIM - SIMPLE<br>RAS - BASIC<br>COM - COMPACT | |
| PT | Prozedurtyp [1] | |
| | PT = d 1 | keine |
| | Prozedur \| d 1<br>AP 62/64 \| H'ØØ'/H'Ø4' | |
| EOF | EOF-Adresse | |
| | EOF = adr<br>Beachte:<br>Indirektadressierung ist nicht erlaubt (at 5) | X'ØØØØ' |
| ERR | ERROR-Adresse | |
| | ERR = adr<br>Beachte:<br>Indirektadressierung ist nicht erlaubt (at 5) | X'ØØØØ' |
| IOA | Adresse des Pufferbereichs für einen Datenblock | |
| | IOA = adr<br>Beachte:<br>Indirektadressierung ist nicht erlaubt (at 5) | X'ØØØØ' |

| | | |
|---|---|---|
| DI | Dateischutzindikator[1] | |
| | SI = si<br>si ::= { COPY / SAVE / DEL / ERASE } | Standard:<br>COPY |
| ORG | Organisation[1] | |
| | ORG = $org_{PD}$ / $org_{KMBG}$<br>$org_{PD}$ ::= { KEY (it,kp,kl,nt) / KEYC(it,kp,kl,lit) / SEQ / DIR / AUTO [(at5$_{abs}$)] }<br>kl ::= l Schlüssellänge<br>kp ::= adr Schlüsselposition<br>it ::= adr Adresse Indextabelle<br>lit ::= adr Länge der Indextabelle<br>Beachte:<br>1. Der Typ des Ausdrucks adr muß für kp und lit absolut sein<br>2. Indirektadressierung ist bei kp, it, lit nicht erlaubt (at 5) | Standard:<br>1. SEQ/SIM |
| | $org_{KMBG}$ = KMBG-Dateistufe<br>$org_{KMBG}$ ::= { SIM / PAS / COM } | |

| | | |
|---|---|---|
| IOS | Länge des Pufferbereichs für einen Datenblock | |
| | IOS = adr<br>Beachte:<br>1. Typ des Ausdrucks adr muß absolut sein<br>2. Indirektadressierung ist nicht erlaubt (at 5) | X'0000' |
| RCS | Maximale Satzlänge | |
| | RCS = adr<br>Beachte:<br>1. Typ des Ausdrucks muß absolut sein<br>2. Indirektadressierung ist nicht erlaubt (at 5) | 128 |
| DA | Logische Geräteadressen | |
| | DA = (da)<br>da ::= g $\left[,g\right]^{*i-1}$,<br>wobei $1 \leq i \leq 4$ | da := 256 |

1) PT und {SI,DT,ORG} sind alternativ anzugeben.
PT gibt den Prozedurtyp für die DFÜ an, SI, DT, ORG charakterisieren die Dateiarbeit mit magnetischen Datenträgern.
Bei Nichtbeachtung erfolgt Fehlerausschrift.

## 5.12. Definition des dynamischen Dateidefinitionsblocks (DCB)

Format:

| Markenfeld | OP-Codefeld | Operandenfeld |
|---|---|---|
| symbol: | DCB | $\bar{I}$ |

Die Pseudoanweisung DCB reserviert 13 + $\bar{I}$ x 13 Bytes, wobei $\bar{I}$ die Anzahl der Geräteadressen ist.

$0 \leq \bar{I} \leq 255$

# 6. Datenarten

## 6.1. Grundsätzliche Bemerkungen

Die in den Abschnitten 6.2. bis 6.4. beschriebenen Datenarten gelten bei der Abarbeitung von Programmen für die Ein- und Ausgabe von Daten mit den Ein- und Ausgabebefehlen sowie für die Verarbeitung. Sie entsprechen den Datenarten für im Programm zu verwendende Konstanten, die mit den unter Abschnitt 5.8. beschriebenen Pseudobefehlen definiert werden. Im Abschnitt 6.5. wird auf die interne Darstellung binärer Daten eingegangen.

## 6.2. Eingabe über Tastatur

Eingaben werden durch das Steuerprogramm SIEX 1526 und den Makrointerpreter MINT 1520 (SIEX) organisiert.

Mit einem Eingabebefehl werden die Daten zeichenweise fortlaufend eingegeben. Die Eingabe wird durch eine Starttaste beendet. Die Eingabelänge ist vom Programmierer frei wählbar.
Die Anfangsadresse adr und die Länge des Datenbereichs l werden vom Programmierer im Eingabebefehl programmiert.
Das Eingeben wird abgebrochen, sobald eine Starttastenbetätigung erkannt wurde. Spätestens aber geschieht das, wenn der interne Datenbereich gefüllt ist (Fehlerlampe leuchtet).

Nicht belegte Stellen werden mit H'Ø' bei gepackten Zahlen und mit H'ØØ' bei ungepackten Zahlen sowie bei Zeichenketten gefüllt.

Mit einem speziellen Eingabebefehl (ENTD, ENTP) ist das Einblenden einer Interpunktionsmaske möglich.

Der Programmierer bestimmt mit der Art des Eingabebefehls das Format:

Eingabe ungepackte Dezimalzahl
Eingabe gepackte Dezimalzahl (BCD-Zahl)
Eingabe alphanumerische Daten (Zeichenkette)

## 6.3. Ausgabe auf Drucker oder Bildschirm

Ausgaben werden durch das Steuerprogramm SIEX 1526 und den Makrointerpreter MINT 1520 (SIEX) organisiert.

Die Ausgabelänge ist vom Programmierer frei wählbar. Die Anfangsadresse adr und die Länge des auszugebenden Bereiches werden vom Programmierer im Ausgabebefehl programmiert. Bei speziellen Ausgabebefehlen (PUT, PUTP) ist das Einblenden einer Maske möglich.

Der Programmierer bestimmt mit der Art des Ausgabebefehls das Format:

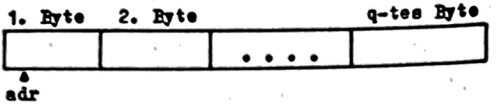
Ausgabe ungepackte Dezimalzahl
Ausgabe gepackte Dezimalzahl (BCD-Zahl)
Ausgabe Zeichenkette

## 6.4. Daten im Hauptspeicher

**Datenfeld:**
Ein Datenfeld wird mit adr und q adressiert, wobei

q ::= 1/k

Es beginnt mit adr als adreßniedrigste Stelle und endet mit adr + q - 1 als adreßhöchste Stelle.




## 6.4.1. Ungepackte Dezimalzahlen

Numerische Werte werden rechtsbündig mit einem Zeichen je Byte in das Datenfeld eingetragen. An adreßhöchster Stelle steht das Vorzeichen oder die wertniedrigste Ziffernstelle (wenn kein Vorzeichen vorhanden ist).

Bei Darstellung mit Vorzeichen sind Zahlen der maximalen Länge von 17 Byte, bei Darstellung ohne Vorzeichen von 16 Byte möglich.

Als positives Vorzeichen wird + (Codierung: H'2B') und Leerzeichen (Codierung: H'20') erkannt. Als negatives Vorzeichen wird - (Codierung: H'2D') erkannt.

Bei numerischen Eingaben wird mit Drücken einer beliebigen Starttaste generell Leerzeichen (d.h. Vorzeichen +) erzeugt. Soll ein negatives Vorzeichen eingetragen werden, so muß vor Betätigen der Starttaste die Taste "-" auf der numerischen Tastatur gedrückt werden.

## 6.4.2. Gepackte Dezimalzahlen

Numerische Werte werden rechtsbündig mit 2 Zeichen je Byte in das Datenfeld eingetragen. In der niederwertigen Tetrade des an adreßhöchster Stelle liegenden Bytes wird das Vorzeichen eingetragen.

Vorzeichenkodierung:

H'C' für +, wenn Taste +
H'F' für +, wenn Leerzeichentaste
H'D' für -

Nach dem Vorzeichen folgen die Ziffernstellen, beginnend mit wertniedrigster Stelle.

Die Feldlängen können nur 4 bzw. 8 Byte betragen, d.h. es sind maximal 7 bzw. 15 Ziffernstellen möglich.

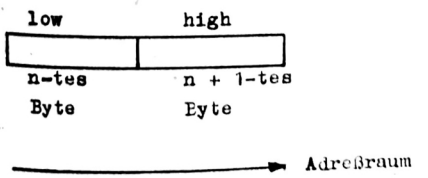
Das Eintragen des Vorzeichencodes erfolgt analog der Beschreibung im Abschnitt 6.4.1.

## 6.4.3. Alphanumerische Daten

Alphanumerische Daten werden linksbündig, d.h. beginnend an der adreßniedrigsten Stelle, in das Datenfeld eingetragen.

## 6.5. Binäre Daten

Binäre Daten, z.B. Adressen, binäre Werte für Binärarithmetik, werden in 2 Byte, zuerst low-Teil, dann high-Teil in aufsteigender Hauptspeicheradreßfolge abgelegt und verarbeitet.



## 7. Assemblerlistenausgabe

Durch den Assembler MEAS 1520 (SIEX) kann wahlweise die Übersetzungs- oder die Fehlerliste ausgegeben werden. Ausgabegeräte sind Drucker oder Bildschirm.

### 7.1. Übersetzungsliste

#### 7.1.1. Form und Inhalt

Die Übersetzungsliste dient zur Überprüfung des Quellprogramms und seiner Übersetzung durch das Übersetzungsprogramm MEAS 1520 (SIEX). Die vollständige Listenzeile ist in 5 Felder unterteilt.

Feld 1: Fehlerkennzeichen
Es dient zur Aufnahme eines 2stelligen Fehlerkennzeichens, womit syntaktische Fehler (Verstöße gegen die Sprechregeln) gekennzeichnet werden (s. Abschnitt 7.1.4.). Es kann nur der erste erkannte Fehler eingetragen werden.

Wenn MEAS 1520 (SIEX) in einer Zeile keine formalen Fehler erkennt, bleibt das Feld 1 leer.

Feld 2: Aktueller Inhalt des Speicherplatzzuweisungszählers
In das Feld 2 wird der aktuelle Wert des Speicherplatzzuweisungszählers als 4stellige Hexadezimalzahl eingetragen.

Feld 3: Übersetzungstext
Das Feld 3 enthält die durch Übersetzung erzeugten Codebytes des jeweiligen Makro- oder Pseudobefehls.

Die Codierungen werden beim Laden des Programms aus der Phasenbibliothek in die entsprechenden Speicherzellen eingetragen. Die Angabe der Codierung erfolgt hexadezimal je Byte.

Es werden maximal 8 Codebytes je Feld ausgegeben. Bei Anweisungen, die mehr als 8 Bytes belegen, wird in der nächsten Zeile mit Feld 2 und 3 fortgesetzt.

Feld 4: Quellzeilenadresse

Feld 4 enthält eine 4stellige hexadezimale Quellzeilennummer. Sie repräsentiert die Adresse der Quellzeile in der Quellbibliothek auf Diskette.

Alle Zeilen des Quellprogramms, auch reine Kommentarzeilen, erhalten eine Quellzeilennummer. Sie wird bei Korrektur des Quellprogramms mit dem Editor EDIT 1520 (SIEX) als Parameter des Korrekturkommandos benötigt.

Feld 5: Quelltext

Das Feld 5 enthält den Quelltext (maximal 60 Zeichen), d.h. den vollständigen Inhalt einer Quellzeile einschließlich Marken-, OP-Code-, Operanden- und Kommentarfeld.

Die Felder werden durch Leerzeichen voneinander getrennt.

### 7.1.2. Ausgabe über Drucker

Jede Übersetzungslistenseite wird mit einem Kopf begonnen. Nach einer Leerzeile folgt die Übersetzungsliste mit der Folge Feld 1, 2, 3, 4, 5.

Die Liste wird abgeschlossen:

a) wenn keine syntaktischen Fehler erkannt wurden mit

   END ASSEMBLY

b) wenn syntaktische Fehler erkannt wurden mit

   ERROR ASSEMBLY

7.1.3.
Ausgabe über Bildschirm

Bei verkürzter Ausgabe der Übersetzungszeile werden nur

Feld 1, Feld 4 und Feld 5 mit Länge 55

ausgegeben.
Bei vollständiger Ausgabe der Übersetzungszeile werden 2 Bildschirmzeilen belegt:

1. Zeile: Feld 4, Feld 5
2. Zeile; Feld 1, Feld 2, Feld 3.

Kopfzeile und Listenabschluß werden, wie in Abschnitt 7.1.2. beschrieben, ausgegeben.

7.1.4.
Fehlerkennzeichen

| Nr. | Bedeutung |
|---|---|
| Ø1 | Nicht zulässiger Operationscode |
| Ø2 | Zeile ohne Zeichen |
| Ø3 | Marke oder OPC nicht mit Buchstaben begonnen |
| Ø4 | Unkorrektes Symbol |
| Ø5 | Unkorrektes Trennzeichen |
| Ø6 | Parameterteil bei Pseudooperation fehlt |
| Ø7 | Doppeldefinition von Symbolen |
| Ø8 | Zahl (z.B. Länge) übersteigt zugelassenen Wertebereich |
| Ø9 | Unkorrekte Darstellung einer Dezimalzahl |
| 1Ø | Unerlaubter Konstantentyp |
| 11 | 1 nicht korrekt |
| 12 | Unkorrekte Zeichenkette C |
| 13 | Zeichenkette größer als Zahl der vereinbarten Zeichen |

| Nr. | Bedeutung |
|---|---|
| 14 | Unkorrekte Zeile (z.B. Zeichen ungleich ' zwischen Konstantendefinition und ; bzw. NL) |
| 15 | Unkorrekte Zeichenkette N |
| 16 | Unkorrekte Zeichenkette (z.B. Apostroph fehlt): allgemein |
| 17 | Unkorrekte Zeichenkette Typ H |
| 18 | Unkorrekte V- oder A-Adreßkonstante |
| 19 | Mehr als 255 externe Bezüge |
| 20 | Fehlerhafter Adreßtyp |
| 21 | Für Adreßtyp nicht zugelassenes Längenattribut angegeben |
| 22 | Kein absoluter Adreßtyp |
| 23 | Absoluter Wert > 255 |
| 24 | Kein darstellbares Alphazeichen (z.B. Bitfehler) |
| 25 | Falscher Adreßausdruck |
| 26 | Verwendung eines nicht definierten Symbols (Marke nicht vorhanden) |
| 27 | Parameter fehlerhaft |
| 28 | - |
| 29 | - |
| 30 | - |
| 31 | - |
| 32 | Pseudobefehl mit Marke, obwohl nicht erlaubt |
| 33 | Nicht markierte EQU-Anweisung |
| 34 | $\overline{1}$ im DCB > 128, d.h. $\overline{1}$ = 4 wurde standardmäßig erzeugt |
| 35 | Nicht zulässiges Schlüsselwort in DDB-Anweisung |
| 36 | Parameterfehler in DDB-Anweisung |

## 7.2. Fehlerliste

Die Ausgabe der Fehlerliste kann wahlweise über Drucker oder Bildschirm erfolgen. Es kommt Feld 1, Feld 4 und Feld 5 für jede fehlerbehaftete Quellzeile zur Angabe.
Bei Bildschirmanzeige wird Feld 5 auf 55 Zeichen beschränkt.

## 8. Beschreibung der Makrobefehle

In diesem Abschnitt sind alle Makrobefehle, die vom SIEX 1520/MINT 1520 (SIEX) abgearbeitet werden, beschrieben. Die Beschreibung enthält

die Befehlssyntax
das physische Befehlsformat (Maschinencodeformat)
die Anzahl der indirekt angebbaren Adressen (Anzahl I-Bits)
die Funktionsdefinition
Bedingungscodes
Erläuterungen zur Befehlsausführung einschließlich der Semantik der Operanden

### Allgemeine Hinweise:

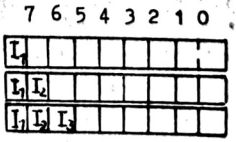
1. In der Befehlssyntaxdarstellung wurde im Namensfeld durchgängig auf die Angabe " [symbol:] " verzichtet, d.h. jeder Befehl ist markierbar.
2. Die Anzahl der I-Bits gibt die Anzahl der möglichen Indirektadreßangaben für die Ausdrücke adr an.

### Beachte:

Ist die Anzahl der I-Bits kleiner als die Anzahl der Adressen im Befehl, so sind die von links gezählten Adreßausdrücke nur indirekt angebbar.

### Darstellung im Maschinencode:

Die I-Bits belegen physisch die höchstwertigen Bits des Befehlscodebytes - oder des Zusatzbytes



In der Regel werden I-Bits dem Befehlscodebyte zugeordnet. Für diesen Fall gilt die in der Beschreibung unter "Anzahl I-Bits" angegebene Zahl für den Befehlscode.

Abweichend davon werden in einigen Befehlen I-Bits im Zusatzbyte (neben anderen Angaben) eingetragen. Für diesen Fall erfolgt jeweils eine Indexierung der Anzahl der I-Bits mit "z", z.B. $2_z$.

Sind I-Bits gesetzt, so ist der eigentliche Befehlscode bei

$I_1 = 1$ um H'8Ø'
$I_2 = 1$ um H'4Ø'
$I_3 = 1$ um H'2Ø'

erhöht. Bei mehreren gesetzten I-Bits ergibt sich die Erhöhung zu:

$I_1 + I_2$ : H'CØ'
$I_1 + I_3$ : H'AØ'
$I_2 + I_3$ : H'6Ø'
$I_1 + I_2 + I_3$ : H'EØ'

3. Die Semantik der Parameter eines Makrobefehls ist aus der jeweiligen Makrobefehlsbeschreibung ersichtlich.

4. In Anlage 1 wird eine Übersicht zur Gruppeneinteilung der Makrobefehle nach Operandentypen gegeben. Sie dient der Orientierung über die Zulässigkeit von Operanden entsprechend des Makrobefehls.

## 8.1. Befehle zur Dezimalarithmetik

Die Arithmetikbefehle verarbeiten ungepackte Daten (Dezimaldaten) bis zu einer maximalen Länge von 17 Byte, wobei das adreßhöchste Byte das Vorzeichen enthält. Wird auf eine Vorzeichenangabe verzichtet, sind Operanden bis zu einer maximalen Länge von 16 Byte möglich.

| | | |
|---|---|---|
| 1 = 1 | 1 | Ziffernstelle |
| 2 ≤ l ≤ 16 | l | Ziffernstellen |
| | l - 1 | Ziffernstellen und Vorzeichen |
| l = 17 | 16 | Ziffernstellen und Vorzeichen |

Verarbeitet werden die Operanden stellenweise von der niederwertigsten Ziffernstelle ($10^0$) im Format des längeren Operanden. Der kürzere Operand geht so in die Verarbeitung ein, als ob er eine entsprechende Anzahl von Vornullen hätte.

Das Vorzeichen wird nach den Regeln der Algebra gebildet und, wenn im Operanden angegeben, im ersten Operanden abgelegt.

Das Resultat steht generell im ersten Operanden. Entstehende Vornullen werden mit dem Füllzeichen NUL (H'ØØ') belegt. Der zweite Operand bleibt unverändert.

Zwischen Vorzeichen und der niederwertigsten Ziffernstelle sowie zwischen den Ziffern sind beliebige Zeichen zugelassen.

Diese werden von der arithmetischen Rechnung ausgelassen. Das Ergebnis erhält jedoch die Formatierung (Maske) des ersten Operanden, wenn also o.g. Zeichen

ein Leerzeichen (␣),
ein Punkt (.) oder
ein Minuszeichen (-) sind,

dann bleiben diese als Maske erhalten. Alle anderen Zeichen werden überschrieben.

| | | | |
|---|---|---|---|
| | | | Add (decimally)<br>Subtract (decimally) |
| Kommando-feld | Operations-feld | Operanden- und Kommentarfeld | A/S |
| | A<br>S | adr1 [(l)], adr2 [(l)] | |

**Befehlsformat:**

| 36 | 11 | 11 | adr1 | 12 | adr2 |
|---|---|---|---|---|---|

Anzahl I-Bit:
2

**Funktion:**
Addition/Subtraktion zweier Dezimalzahlen

**Bedingungscode:**

0 Ergebnis = 0
1 Ergebnis < 0
2 Ergebnis > 0
3 Überlauf

Der zweite Operand wird zum/vom ersten Operanden addiert/subtrahiert und das Ergebnis an der Stelle des ersten Operanden abgespeichert.

Ein Überlauf liegt vor, wenn der erste Operand zu kurz ist, um alle Ziffern der Summe (außer Vornullen) aufzunehmen.

| | | | Multiply (decimally)<br>Divide (decimally) |
|---|---|---|---|
| Namens-feld | Operations-feld | Operanden- und Kommentarfeld | |
| | M<br>D | n,adr1[(1)],adr2[(1)] | |

**Befehlsformat:**

12

| 13 | 1 1 | adr 1 | 1 2 | adr 2 | n |
|---|---|---|---|---|---|

Anzahl I-Bit: 2

**Funktion:**

Multiplikation/Division zweier Dezimalzahlen

**Bedingungscode:**

| | |
|---|---|
| Ø | Ergebnis = Ø |
| 1 | Ergebnis < Ø |
| 2 | Ergebnis > Ø |
| 3 | Überlauf |

Der Inhalt des ersten Operanden wird mit dem/durch den Inhalt des zweiten Operanden multipliziert/dividiert.

Das Ergebnis wird bei der Multiplikation um n Stellen nach rechts verschoben (Diese Stellen gehen verloren). Bei der Division wird das Ergebnis um n Stellen nach links verschoben (richtiger: Die Division wird weitergeführt). Das Ergebnis wird anschließend rechtsbündig ab Adresse adr 1 eingeschrieben.

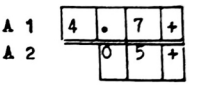
Die über l 1 hinausgehenden Stellen gehen verloren. In diesem Fall wird nur Überlauf angezeigt (BC 3), d.h. das Überlaufbit hat immer Priorität.

Ist der Divisor gleich Null, wird keine Division ausgeführt. Wurde vorher eine ON-Bedingung (Zero-Devide) programmiert, so wird zur angegebenen Adresse verzweigt, anderenfalls wird im Programm fortgesetzt (Bedingungscode 3 - wird gesetzt).

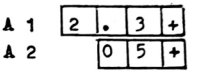
**Beispiele:**

M 1, A 1 (4), A 2 (3)

Vor der Operation:



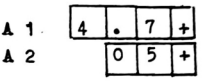

Nach der Operation:



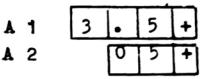

BC 2 gesetzt

M Ø, A 1 (4), A 2 (3)

Vor der Operation:



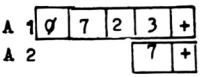

Nach der Operation:



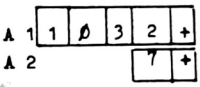

BC 3 gesetzt

D 1, A 1 (5), A 2 (2)

Vor der Operation:

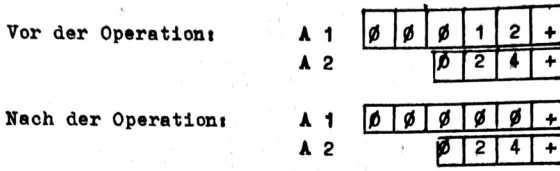



Nach der Operation:

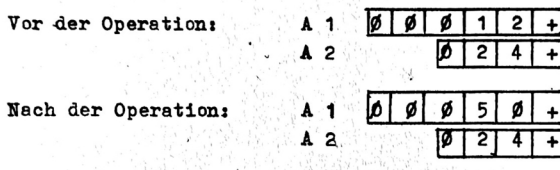



BC 2 gesetzt

D Ø, A 1 (6), A 2 (4)




D 2, A 1 (6), A 2 (4)




## 8.2. Logische Befehle

Die beiden Operanden werden als vorzeichenlose logische Größen aufgefaßt und durch die Funktionen "UND" ($\wedge$), "ODER" ($\vee$) bzw. "EXCLUSIVE ODER" ($\oplus$) bitweise miteinander verknüpft.

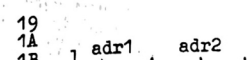
| a | b | a $\wedge$ b | a $\vee$ b | a $\oplus$ b |
|---|---|---|---|---|
| Ø | Ø | Ø | Ø | Ø |
| Ø | 1 | Ø | 1 | 1 |
| 1 | Ø | Ø | 1 | 1 |
| 1 | 1 | 1 | 1 | Ø |

Das Ergebnis der Operation überschreibt den ersten Operanden. Der zweite Operand bleibt unverändert.

Die Anzahl der zu verknüpfenden Bytes ist variabel (max. 256), und wird durch 1 spezifiziert bzw. durch die Länge des ersten Operanden.

| | | | |
|---|---|---|---|
| | | | And / Or / Exclusive or |
| Namens-feld | Operations-feld | Operanden- und Kommentarfeld | N/O/X |
| | N<br>O<br>X | adr 1[1], adr 2 | |

**Befehlsvormat:**




**Anzahl-I-Bit:**

2

**Funktion:**

Bitweise Verknüpfung von Zeichenketten entsprechend den logischen Funktionen "UND", "ODER", "EXCLUSIVE ODER"

**Bedingungscode:**

Ø wenn alle Ergebnisbits = Ø
1 sonst

**Beispiel:**

Löschen des Operanden FELD der Länge 3 Byte

```
X   FELD (3), FELD
```

| | vorher | | | nachher |
|---|---|---|---|---|
| FELD | 1ØØØ1Ø11 | ØØ1Ø1Ø1Ø | ØØØØ1111 | ØØØØØØØØ ØØØØØØØØ ØØØØØØØØ |
| FELD | 1ØØØ1Ø11 | 00101010 | 00001111 | |

BC Ø gesetzt

## 8.3. BCD-Arithmetik

Die BCD-Befehle verarbeiten Operanden im gepackten Format der Länge 4 oder 8 Byte.

Die Verarbeitung der Operanden erfolgt byteweise von rechts nach links in der Länge des längeren Operanden.

Der kürzere Operand wird so verarbeitet, als ob er führende Nullen hätte.

Das Vorzeichen wird nach den Regeln der Algebra gebildet.
Codierung der Vorzeichentetrade:

H'C' für +
H'F' für +
H'D' für -

Ein Ergebnis von Null ist stets positiv.
Der zweite Operand bleibt unverändert.

| | | | Add decimal packed/ Subtract decimal packed |
|---|---|---|---|
| Namens-feld | Operations-feld | Operanden- und Kommentarfeld | AP/SP |
| | AP<br>SP | adr1[(k1)],adr2[(k2)] | |
| **Befehlsformat:**<br>3 A p 1<br>3 A p 3 adr 1 adr 2 | | | Anzahl I-Bit:<br>2 |
| **Funktion:**<br>Addition / Subtraktion zweier gepackter Dezimalzahlen | | | |

Bedingungscode:

| | | | |
|---|---|---|---|
| Ø | Ergebnis = Ø | 2 | Ergebnis > Ø |
| 1 | Ergebnis < Ø | 3 | Überlauf |

Der zweite Operand wird zum/vom ersten Operanden addiert/subtrahiert und das Ergebnis im ersten Operanden gespeichert.

Ein Überlauf liegt vor, wenn der erste Operand zu kurz ist, um alle Ziffern der Summe (außer Vornullen) aufzunehmen.

Belegung des p 1-, p 3-Bytes:

| | | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | |
|---|---|---|---|---|---|---|---|---|---|---|
| p | 1 | ki | kj | Ø | Ø | Ø | Ø | Ø | Ø | ki Längenangabe 1. Operand |
| p | 3 | ki | kj | Ø | 1 | Ø | Ø | Ø | Ø | kj Längenangabe 2. Operand |

ki, kj = Ø bei Länge k = 4
1 bei Länge k = 8

| Namensfeld | Operationsfeld | Operanden- und Kommentarfeld | Add decimal packed reverse/ Subtract decimal packed reverse |
|---|---|---|---|
| | APR<br>SPR | adr1[(k1)],adr2[(k2)] | APR/SPR |

Befehlsformat:

3 A p 2

| 3 A | p 4 | adr 1 | adr 2 |
|---|---|---|---|

Anzahl I-Bit:
2

Funktion:
Subtraktion/Addition zweier gepackter Dezimalzahlen bei gesetztem KØ-Bit

Bedingungscode:

| | |
|---|---|
| Ø | Ergebnis = Ø |
| 1 | Ergebnis < Ø |
| 2 | Ergebnis > Ø |
| 3 | Überlauf |

Ist das KØ-Bit (Selektor Ø - Stornoselektor - ein, d.h. Setzen Bit Ø des K-Registers) gesetzt, so bedeutet dies, daß die Addition/Subtraktion in ihre Umkehroperation gewandelt wird. Die Wirkungsweise des Befehls APR/SPR entspricht dann der Wirkungsweise des SP/AP-Befehls.

Wenn das KØ-Bit gelöscht ist, entspricht die Wirkungsweise des APR-/SPR-Befehls der Wirkungsweise des AP-/SP-Befehls.

Aufbau des p 2-, p 4-Bytes:

| | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 |
|---|---|---|---|---|---|---|---|---|
| p 2 | ki | kj | Ø | Ø | 1 | Ø | Ø | Ø |
| p 4 | ki | kj | Ø | 1 | 1 | Ø | Ø | Ø |

ki Längenangabe 1. Operand
kj Längenangabe 2. Operand

ki, kj = { Ø bei Länge k = 4
1 bei Länge k = 8

| Namens-feld | Operations-feld | Operanden-, und Kommentarfeld | Multiply decimal packed / Divide decimal packed |
|---|---|---|---|
| | MP<br>DP | n,adr1 [(k1)] ,adr2 [(k2)] | MP/DP |

**Befehlsformat:**

3 A p 5

| 3 A | p 6 | adr 1 | adr 2 |
|---|---|---|---|

Anzahl I-Bit:

2

**Funktion:**

Multiplikation/Division zweier gepackter Dezimalzahlen

**Bedingungscode:**

Ø Ergebnis = Ø
1 Ergebnis < Ø
2 Ergebnis > Ø
3 Überlauf

Der Inhalt des ersten Operanden wird mit/durch den Inhalt des zweiten Operanden multipliziert/dividiert. (Die Division erfolgt mit Rundung. Es wird aufgerundet, wenn die Rundungsstelle $\geq 5$ ist.) Das Ergebnis wird im ersten Operanden gespeichert.

**Multiplikation:**

Das Ergebnis wird bei der Multiplikation um n Stellen nach rechts verschoben (Diese Stellen gehen verloren).

Bei der Stellenabstreichung wird gerundet, Rundungsziffer ist die werthöchste abzustreichende Ziffer.
Die erforderliche Stellenabstreichung n (Ø - n - 15) ist nach folgender Formel zu ermitteln:

$$n = N_{MD} + N_{MR} - N_{PR}$$

N Anzahl der Kommastellen
MD Multiplikand
MR Multiplikator
PR Produkt

Überlauf wird angezeigt, wenn der erste Operand zu kurz ist, um alle Ziffern des Produkts (außer Vornullen) aufzunehmen.

Division:

Bei der Division wird das Ergebnis um n Stellen nach links verschoben, d.h. die Division wird um n Stellen nach dem Komma weitergeführt. Das Ergebnis wird anschließend rechtsbündig ab Adresse adr 1 eingeschrieben. Überschreitet das Ergebnis bei Stellenverschiebung die Länge des ersten Operanden, erfolgt eine Überlaufanzeige über den Bedingungscode.

Die Stellenverschiebung n ($\emptyset \leq n \leq 15$) wird nach folgender Formel berechnet:

$$n = N_{QU} + N_{DR} + 3 - N_{DD}$$

| | |
|---|---|
| N | Anzahl der Kommastellen |
| QU | Quotient |
| DR | Divisor |
| DD | Dividend |

Ist der Divisor gleich Null, wird keine Division ausgeführt. Wurde vorher eine ON-Bedingung (Zero-Divide) programmiert, so wird zur angegebenen Adresse verzweigt. Andernfalls wird im Programm fortgesetzt (Bedingungscode 3 - Überlauf).

Aufbau des p 5-/p 6-Bytes:

| | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 |
|---|---|---|---|---|---|---|---|---|
| p 5 | ki | kj | 1 | Ø | ← | n | | → |
| p 6 | ki | kj | 1 | 1 | ← | n | | → |

ki Längenangabe 1. Operand
kj Längenangabe 2. Operand

ki, kj = { Ø, bei Länge k = 1
1, bei Länge k = 8

| Namensfeld | Operationsfeld | Operanden- und Kommentarfeld |
|---|---|---|
| | CP | adr1[(k1)],adr2[(k2)] |

Compare decimal packed

C P

**Befehlsformat:**

| 3 A | p 7 | adr 1 | adr 2 |
|---|---|---|---|

Anzahl I-Bit: 2

**Funktion:**

Vorzeichengerechter Vergleich zweier gepackter Dezimalzahlen

**Bedingungscode:**

| | |
|---|---|
| Ø | 1. Operand = 2. Operand |
| 1 | 1. Operand < 2. Operand |
| 2 | 1. Operand > 2. Operand |

Der Inhalt des ersten Operanden wird mit dem Inhalt des zweiten verglichen.
Das Vergleichsergebnis, das sich immer auf den ersten Operanden bezieht, wird im Bedingungscode angezeigt.

Der Vergleich erfolgt algebraisch (vorzeichengerecht). Negative Zahlen sind immer kleiner als positive, unabhängig von der Verschlüsselung der Vorzeichen.
Ausnahme: "Plus Null" ist gleich "Minus Null"!

Aufbau des p 7-Bytes:

| | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 |
|---|---|---|---|---|---|---|---|---|
| p 7 | ki | kj | Ø | 1 | Ø | 1 | Ø | Ø |

ki Längenangabe 1. Operand
kj Längenangabe 2. Operand

ki, kj = { Ø bei Länge k = 4
1 bei Länge k = 8

| Namensfeld | Operationsfeld | Operanden- und Kommentarfeld |
|---|---|---|
| | SL<br>SR | n, adr [(k)] |

Shift left:
Shift right:

SL/SR

**Befehlsformat:**

3 C p 8
3 C | p 9 | adr |

Anzahl Z-Bit:
1

**Funktion:**

Verschiebung einer gepackten Dezimalzahl um n Stellen (Tetraden) nach links/rechts

**Bedingungscode:**

0 Ergebnis = 0
1 Ergebnis < 0
2 Ergebnis > 0

Der Operand wird um n ($0 \leq n \leq 15$) Stellen nach links/rechts verschoben, wobei die hinausgeschobenen Stellen nicht geprüft werden und verloren gehen.

Die freiwerdenden Stellen werden mit H'0' aufgefüllt.

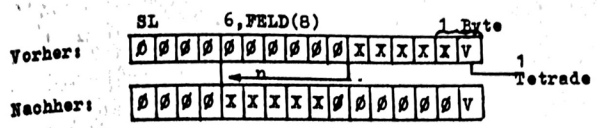
Die Vorzeichenstelle wird nicht mit verschoben.

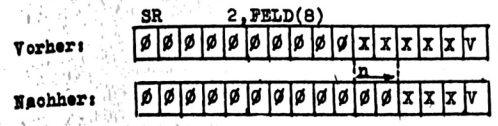
Aufbau des p 8-/p 9-Bytes:

| | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 |
|---|---|---|---|---|---|---|---|---|
| p 8 | ki | 0 | 0 | 0 | ←——n——→ | | | |
| p 9 | ki | 0 | 0 | 1 | ←——n——→ | | | |

$$ki = \begin{cases} 0, & \text{bei } k = 4 \\ 1, & \text{bei } k = 8 \end{cases}$$

**Beispiele:**


SL 6,FELD(8)
1 Byte
Vorher:
Ø Ø Ø Ø Ø Ø Ø Ø Ø Ø Ø X X X X X V
n
1 Tetrade
Nachher:
Ø Ø Ø Ø X X X X X Ø Ø Ø Ø Ø Ø V


SR 2,FELD(8)
Vorher:
Ø Ø Ø Ø Ø Ø Ø Ø Ø Ø Ø X X X X X V
n
Nachher:
Ø Ø Ø Ø Ø Ø Ø Ø Ø Ø Ø Ø X X X V

## 8.4.
## Befehle zur binären Verarbeitung

Die Operanden sind vorzeichenlose Größen.
Sie haben eine feste Länge von 2 Byte.
Das Ergebnis der Operation wird im ersten Operanden gespeichert, der zweite Operand bleibt unverändert.

| Namensfeld | Operationsfeld | Operanden- und Kommentarfeld |
|---|---|---|
| | AB<br>SB<br>MB | adr 1, adr 2 |

Add binary/
Subtract binary/
Multiply binary

AB/SB/MB

**Befehlsvormat:**

| 30 | | |
|---|---|---|
| 31 | | |
| 32 | adr 1 | adr 2 |

Anzahl I-Bit:
2

**Funktion:**

Binäre Addition/Subtraktion/Multiplikation von zwei Operanden

**Bedingungscode:**

Ø Ergebnis = Ø
2 Ergebnis > Ø
3 Überlauf

AB/SB bewirkt, daß zum/vom ersten Operanden der zweite Operand addiert/subtrahiert wird.
MB bewirkt die Multiplikation beider Operanden.

**Beispiel:**

AB FELD 1, FELD 2

| | | L | | H | | |
|---|---|---|---|---|---|---|
| Vor der Operation: | FELD 1 | E | 1 | Ø | 1 | 481 |
| | | | | + | | + |
| | FELD 2 | 3 | C | Ø | Ø | 6Ø |
| Nach der Operation: | FELD 1 | 1 | D | Ø | 2 | 541 |
| | FELD 2 | 3 | C | Ø | Ø | 6Ø |

BC 2 gesetzt

| Namensfeld | Operationsfeld | Operanden- und Kommentarfeld |
|---|---|---|
| | ABI | adr, d 2 |

Add binary immediate

ABI

**Befehlsformat:**

65 | d,2 | adr

Anzahl L-Bit:

1

**Funktion:**

Addition eines im Befehl angegebenen 2-Byte-Direktoperanden zum ersten Operanden

**Bedingungscode:**

Ø Ergebnis = Ø
2 Ergebnis > Ø
3 Überlauf

Der zweite Operand ist ein 2-Byte-Direktoperand.

Beispiel:

```
AEI   FELD,H'7302'
Vor der Operation: FELD
                   d 2
Nach der Operation: FELD
BC 2 gesetzt
```

| L | | H | |
|---|---|---|---|
| 9 | 7 | 1 | 9 |
| 7 | 3 | 8 | 2 |
| Ø | A | 9 | 0 |

| Namensfeld | Operationsfeld | Operanden- und Kommentarfeld |
|---|---|---|
| | MVBI | adr, d 2 |

Move binary immediate

MVBI

Befehlsformat:

64 | d,2 | adr

Anzahl L-Bit:

1

Funktion:

Transport eines im Befehl angegebenen 2-Byte-Direktoperanden an die Speicherstellen des ersten Operanden

Bedingungscode:

[-]

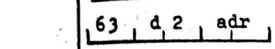
Der zweite Operand ist ein 2-Byte-Direktoperand.

| Namensfeld | Operationsfeld | Operanden- und Kommentarfeld |
|---|---|---|
| | CB<br>CBI | adr 1, adr 2<br>adr, d 2 |

Compare binary/
Compare binary immediate

CB/CBI

**Befehlsformat:**




Anzahl I-Bit:
2/1

**Funktion:**

Binärer Vergleich zweier Operanden

**Bedingungscode:**

Ø Beide Operanden sind gleich.
1 1. Operand < 2. Operand
2 1. Operand > 2. Operand

Bei CBI ist der zweite Operand ein 2-Byte-Direktoperand.

**Beispiel:**

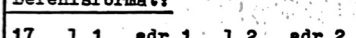
```
CB    FELD 1, FELD 2
```

## 8.5. Transportbefehle

Der zweite Operand überschreibt den ersten Operanden.
Der bisherige Inhalt des ersten Operanden wird zerstört.
Der Inhalt des zweiten Operanden bleibt unverändert.

| Namensfeld | Operationsfeld | Operanden- und Kommentarfeld | | Move |
|---|---|---|---|---|
| | MV | adr1[(l1)], adr2[(l2)] | | MV |

**Befehlsformat:**




**Anzahl I-Bit:** 2

**Funktion:**

Transport von einem Feld in ein anderes, rechts beginnend

**Bedingungscode:**

3 Überlauf (l 2 > l 1)

l 2 > l 1: Es werden l 1 Zeichen übertragen
l 1 ≥ l 2: Es werden l 2 Zeichen übertragen. Die restlichen Stellen des ersten Operanden werden mit "NUL" aufgefüllt (Löschfunktion).

**Beispiel:**

MV FELD 1 (6), FELD 2 (4)

Vorher: FELD 1

FELD 2

Nachher: FELD 1

FELD 2

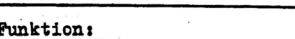
| | | | | | |
|---|---|---|---|---|---|
| 1 | A | , | 2 | + | B |
| | | K | 1 | - | L |
| NUL | NUL | K | 1 | - | L |
| | | K | 1 | - | L |

| Namens-feld | Operations-feld | Operanden- und Kommentarfeld |
|---|---|---|
| | MVC | adr 1 [(l)], adr 2 |

Move character

MVC

**Befehlsformat:**




Anzahl I-Bit: 2

**Funktion:**

Transport von einem Feld in ein anderes, links beginnend

**Bedingungscode:**

-

Der zweite Operand wird an die Speicherstellen (Anzahl l) des ersten Operanden übertragen.

**Move character multiple**

| Namensfeld | Operationsfeld | Operanden- und Kommentarfeld |
|---|---|---|
| | MVM | adr1[(l1)],adr2[(l2)] |

MVM

**Befehlsformat:**

| 16 | l 1 | adr 1 | l 2 | adr 2 |
|---|---|---|---|---|

**Anzahl I-Bit:** 2

**Funktion:**
Transport mehrfach links beginnend

**Bedingungscode:**
3 Überlauf

Die l 2 Zeichen des zweiten Operanden überschreiben den ersten Operanden, links beginnend, mehrfach.

Die Wiederholung erfolgt, bis l 1 Zeichen übertragen werden sind.

BC 3 wird gesetzt, wenn l 1 kein ganzzahliges Vielfaches von l 2 ist.

**Beispiel:**

MVM FELD 1 (9), FELD 2 (4)

Vorher: FELD 1

| 1 | A | . | 2 | 5 | - | M | F | G |
|---|---|---|---|---|---|---|---|---|
| 5 | . | 8 | M | 5 | - | M | F | G |
| 5 | . | 8 | M | 5 | . | 8 | M | 8 |
| 5 | . | 8 | M | 5 | . | 8 | M | 5 |

Nachher: FELD 1

| 5 | . | 8 | M | 5 | . | 8 | M | 5 |
|---|---|---|---|---|---|---|---|---|

BC 3 gesetzt

FELD 2

| 5 | . | 8 | M |
|---|---|---|---|

1. Übertragung

2. Übertragung

3. Übertragung

FELD 2

| 5 | . | 8 | M |
|---|---|---|---|

| Namensfeld | Operationsfeld | Operanden- und Kommentarfeld |
|---|---|---|
| | MVI | adr, d 1 |

Move immediate

MVI

**Befehlsformat:**

| 60 | d 1 | adr |
|---|---|---|

Anzahl I-Bit: 1

**Funktion:**

Transport eines im Befehl angegebenen Direktoperanden in ein Byte im Speicher

**Bedingungscode:**

-

Der zweite Operand ist ein Direktoperand (1 Byte).

## 8.6. Vergleichsbefehle

Der Inhalt beider Operanden bleibt unverändert.
Das Vergleichsergebnis wird im Bedingungscode angezeigt.

| Namensfeld | Operationsfeld | Operanden- und Kommentarfeld | Compare (decimal unpacked) |
|---|---|---|---|
| | C | adr 1 [(l 1)], adr 2 [(l 2)] | C |

**Befehlsformat:**

| 14 | l 1 | adr 1 | l 2 | adr 2 |
|---|---|---|---|---|

Anzahl I-Bit: 2

**Funktion:**

Numerischer Vergleich zweier maskierter Operanden variabler Länge

**Bedingungscode:**

Ø beide Operanden sind numerisch gleich
1 1. Operend < 2. Operand
2 1. Operand > 2. Operand

Von den Zeichenketten werden nur Vorzeichen und Zifferncode entsprechend der Dezimalarithmetik verglichen.
Masken werden übergangen.

**Beispiel:**

C FELD 1 (7), FELD 2 (6)

FELD 1

| a | 1 | 2 | B | , | 1 | 6 |
|---|---|---|---|---|---|---|

\- FELD 2

| 1 | 2 | C | - | 1 | + |
|---|---|---|---|---|---|

EC Ø gesetzt

Verarbeitet werden 16 Ziffern + Vz

| Namens-feld | Operations-feld | Operanden- und Kommentarfeld | Compare logical characters |
|---|---|---|---|
| | CLC | adr 1 [(l)], adr 2 | CLC |

**Befehlsformat:**

| 1 C | l | adr 1 | adr 2 |
|---|---|---|---|

Anzahl L-Zit: 2

**Funktion:**

Logischer Vergleich zweier Operanden links beginnend

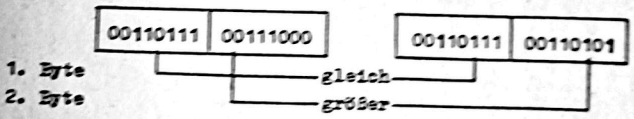
**Bedingungscode:**

0 Beide Operanden sind gleich
1 1. Operand < 2. Operand
2 1. Operand > 2. Operand

Verglichen wird in der Länge des ersten Operanden bzw. in der programmierten Länge l.

Beispiel:

```
CLC   FELD 1 (2), FELD 2
```




Gesamtergebnis: 1. Operand größer 2. Operand
BC 2 gesetzt

| Namens-feld | Operations-feld | Operanden- und Komentarfeld |
|---|---|---|
| | CLI | adr, d 1 |

Compare logical

CLI

**Befehlsformat:**

61 | d 1 | adr

Anzahl I-Bit: 1

**Funktion:**

Vergleich eines Bytes im Speicher mit einem im Befehl angegebenen Direktoperanden

**Bedingungscode:**

Ø gleich
1 kleiner als Direktoperand
2 größer als Direktoperand

| Namens-feld | Operations-feld | Operanden- und Kommenterfeld |
|---|---|---|
| | TM | adr 1, d 1 |

**Test under Mask**

TM

**Befehlsformat:**

| 62 | d 1 | adr |
|---|---|---|

**Anzahl I-Bit:**

1

**Funktion:**

Prüfen der Bit-Konfiguration eines Bytes

**Bedingungscode:**

Ø Alle zu prüfenden Bits sind Null, oder alle Maskenbits sind Null
1 Die zu prüfenden Bits haben die Werte Ø und 1
3 Alle zu prüfenden Bits sind 1

Der Direktoperand stellt eine Maske dar.
Geprüft werden nur die Bits des ersten Operanden, deren korrespondierende Bits in der Maske mit 1 angegeben wurden.

**Beispiel:**

```
TM   FELD 1, H'3 A'
.    .
.    .
.    .
FELD 1: DC  H' D 2'
FELD 1:  11Ø1ØØ1Ø
Maske:   ØØ111Ø1Ø

Ergebnis: Ø1Ø 1
BC 1 gesetzt
```

## 8.7. Konvertierbefehle

| Namensfeld | Operationsfeld | Operanden- und Kommentarfeld | Convert |
|---|---|---|---|
| | CV | adr 1, adr 2 [(l)] | CV |

**Befehlsformat:**

| 34 | l | adr 1 | adr 2 |
|---|---|---|---|

Anzahl I-Bit: 2

**Funktion:**

Konvertieren ungepackte Dezimalzahl ⟶ Binärzahl

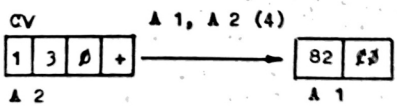
**Bedingungscode:**

Ø Wert = Ø
2 Wert > Ø (hier besser: ≠ Ø)
3 Überlauf

Der zweite Operand ist eine Dezimalzahl und erfordert unbedingt ein Vorzeichen.

Der erste Operand ist eine Binärzahl. Stellt adr2 ein externes Symbol dar, dann ist die Länge immer explizit anzugeben.
Der zweite Operand wird in eine Binärzahl gewandelt und an die Stellen des ersten Operanden transportiert.
Falls die Dezimalzahl größer als $2^{16}-1$ (65535) bzw. negativ ist, so wird BC 3 (Überlauf) angezeigt.

**Beispiel:**




BC 2 gesetzt

| Namensfeld | Operationsfeld | Operanden- und Kommentarfeld |
|---|---|---|
| | CVC | adr 1 [(l)], adr 2 |

Convert to character

CVC

**Befehlsformat:**

| 35 | 1 | adr 1 | adr 2 |
|---|---|---|---|

Anzahl I-Bit: 2

**Funktion:**

Konvertieren Binärzahl ⟶ Ungepackte Dezimalzahl

**Bedingungscode:**

0 Wert = 0
2 Wert > 0
3 Überlauf

Der zweite Operand ist eine Binärzahl.

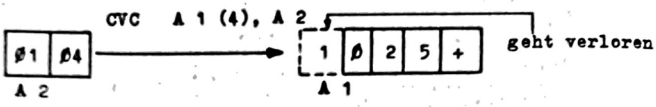
Der erste Operand ist eine Dezimalzahl.

Die Binärzahl wird in eine Dezimalzahl gewendelt und an die Stellen des ersten Operanden transportiert. Es wird immer ein positives Vorzeichen erzeugt (+).

Falls die angegebene Länge des ersten Operanden nicht ausreicht, wird im BC Überlauf angezeigt.

Es wird bis l mit NUL aufgefüllt, falls die Länge der Dezimalzahl kleiner als l - 1 ist.

Beispiel:




BC 3 gesetzt

| Namensfeld | Operationsfeld | Operanden- und Kommentarfeld |
|---|---|---|
| | CVB | adr 1, adr 2 [(k)] |

Convert to binary

CVB

Befehlsformat:

| 38 | p 11 | adr 1 | adr 2 |
|---|---|---|---|

Anzahl I-Bit: 2

Funktion:

Konvertieren gepackte Dezimalzahl → Binärzahl

Bedingungscode:

Ø Ergebnis = Ø
2 Ergebnis ≠ Ø
3 Überlauf

Der zweite Operand muß eine gepackte Dezimalzahl (entsprechend ECD-Arithmetik) sein.

Der erste Operand ist eine Binärzahl (entsprechend binärer Vererteitung).

Der zweite Operand wird in eine Binärzahl gewandelt und an die Stellen des ersten Operanden transportiert.

Ist der Wert der Dezimalzahl größer als $2^{16}-1$ (65535), tritt die Bedingung Überlauf ein (Bedingungscode 3). Die Dezimalzahl wird nicht konvertiert.

Ist die Dezimalzahl negativ, tritt die Bedingung Überlauf ein (Bedingungscode 3 wird gesetzt). Die Dezimalzahl wird konvertiert.

Ist der zweite Operand keine gepackte Dezimalzahl, wird "Datenfehler" angezeigt (Systemfehler Ø7).

Aufbau des p 11-Bytes:

| | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 |
|---|---|---|---|---|---|---|---|---|
| p 11 | ki | Ø | Ø | Ø | Ø | Ø | Ø | Ø |

$$ki = \begin{cases} Ø, & \text{bei } k = 4 \\ 1, & \text{bei } k = 8 \end{cases}$$

Beispiel:

```
CVB     ADDR 1, BCDW (4)
```

| BCDW | ØØ | ØØ | 28 | 7 D |
|---|---|---|---|---|

→ Negatives Vorzeichen

L-Teil, H-Teil

| ADDR 1 | 1F | Ø1 |
|---|---|---|

BC = 3 (Überlauf)

| Namensfeld | Operationsfeld | Operanden- und Kommentarfeld |
|---|---|---|
| | CVD | adr 1 [(k)], adr 2 |

Convert to decimal

CVD

**Befehlsformat:**

| 39 | p 11 | adr 1 | adr 2 |
|---|---|---|---|

**Anzahl L-Bit:**

2

**Funktion:**

Konvertieren Binärzahl ⟶ Gepackte Dezimalzahl

**Bedingungscode:**

Ø Ergebnis = Ø
2 Ergebnis > Ø

Der zweite Operand ist eine Binärzahl (entsprechend binärer Verarbeitung).

Der erste Operand stellt eine gepackte Dezimalzahl (entsprechend BCD-Arithmetik) dar.

Die Binärzahl wird in das BCD-Format entsprechend der programmierten Länge 4 oder 8 Byte überführt. Das Vorzeichen wird dabei immer positiv. Nach links wird mit H'Ø' aufgefüllt.

Aufbau des p 11-Bytes (s. CVB-Befehl)

**Beispiel:**

CVD BCDW (4), ADDR 1

| | L-Teil | H-Teil | | |
|---|---|---|---|---|
| ADDR 1 | 1 F | Ø 1 | | |
| BCDW | ØØ | ØØ | 28 | 7C |

| Namens-feld | Operations-feld | Operanden- und Kommentarfeld |
|---|---|---|
| | PACK | adr 1 [(k)] , adr 2 [(l)] |

Pack

PACK

**Befehlsformat:**

| 37 | p 12 | adr 1 | adr 2 |
|---|---|---|---|

Anzahl I-Bit: 2

**Funktion:**

Konvertieren ungepackte Dezimalzahl ⟶ Gepackte Dezimalzahl

**Bedingungscode:**

-

Das Byte mit der höchsten Adresse der ungepackten Dezimalzahl wird als Vorzeichenbyte betrachtet.

Ist dieses Byte C'-' (H'2 D'), so erhält die gepackte Dezimalzahl ein negatives Vorzeichen (H'D').

Ist dieses Byte C'+' (H'2 B'), so erhält die gepackte Dezimalzahl ein positives Vorzeichen (H'F').

Wird kein gültiges Vorzeichen gefunden, so erhält die gepackte Dezimalzahl ein positives Vorzeichen.

Die Zifferntetraden werden von rechts nach links vom ungepackten Operanden in den gepackten Operanden übernommen, jedoch nur dann, wenn das ungepackte Zeichen eine Ziffer von Ø ... 9 im KOI-7-Code darstellt (H'3Ø' ... H'39'). Anderenfalls wird das Zeichen ignoriert.

Diese Operation wird mit den l Bytes des ungepackten Operanden durchgeführt (l - eingeschränkte Längenangabe, $1 \leq l \leq 64$).

Wenn notwendig, wird der gepackte Operand mit H'Ø' aufgefüllt.

Können nicht alle Ziffern übertragen werden, da der erste Operand zu klein ist, gehen die überzähligen Stellen verloren.

Aufbau des p 12-Bytes:

| | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 |
|---|---|---|---|---|---|---|---|---|
| 12 | ki | l | ← —— l —— → | | | | | |

$ki = \begin{cases} 0, & \text{bei } k = 4 \\ 1, & \text{bei } k = 8 \end{cases}$

l = $000001_B$ = 1 Stelle
l = $111111_B$ = 63 Stellen
l = $000000_B$ = 64 Stellen

Beispiele:

1. PACK SUM (8), CHAR (16)

PACK

CHAR

| 30 | 31 | 32 | 33 | 34 | 35 | 36 | 37 | 38 | 39 | 30 | 31 | 32 | 33 | 34 | 35 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|

Geht verloren

Kein gültiges Vorzeichen

SUM

| 12 | 34 | 56 | 78 | 90 | 12 | 34 | 5+ |
|---|---|---|---|---|---|---|---|

2. PACK FELD 1 (4), EIN

EIN

| 37 | 76 | 38 | 39 | 2D |
|---|---|---|---|---|

wird ignoriert

FELD 1

| 00 | 00 | 78 | 9 D |
|---|---|---|---|

| Namensfeld | Operationsfeld | Operanden- und Kommentarfeld | Unpack |
|---|---|---|---|
| | UNPK | m, adr 1 [(l)], adr 2 [(k)] | UNPK |

**Befehlsformat:**

| 36 | p 12 | adr 1 | adr 2 | m |
|---|---|---|---|---|

**Anzahl I-Bit:** 2

**Funktion:**

Konvertieren gepackte Dezimalzahl ⟶ ungepackte Dezimalzahl

**Bedingungscode:**

[-]

Ist die gepackte Dezimalzahl positiv, so erhält der erste Operand rechtsbündig ein Space (H'2Ø').

Stellt der zweite Operand eine negative Dezimalzahl dar, so wird in den ersten Operanden rechtsbündig ein Minuszeichen (H'2 D') eingetragen.

Zwischen jede Zifferntetrade des zweiten Operanden werden bei der Übertragung in den ersten Operanden Tetraden mit der Codierung H'3' eingefügt, so daß sich Zahlen im ungepackten Format (KOI-7-Code) ergeben.

Durch Angabe einer Maske m über das Maskenbyte ist eine Druckaufbereitung möglich.

Die Maskenbits besitzen folgende Bedeutung:

| Bit | 7 6 5 | 4 | 3 2 | 1 Ø |
|---|---|---|---|---|
| | Hunderterinterpunktion, Tausenderinterpunktion | Dezimalstellenabtrenn. | Führende Stellen | Dezimalstellen |
| | ØØØ = NI (ohne) | Ø = Comma | ØØ = Zero (Z) | ØØ = Ø Stellen<br>Ø1 = 1 Stelle |
| | 11Ø = HP (Point) | | | |
| | 111 = HS (SPACE) | (C) | Ø1 = SPACE (S) | 1Ø = 2 Stellen |
| | 1ØØ = TP (Point) | 1 = Point | 1Ø = Stern | |
| | 1Ø1 = TS (SPACE) | (P) | (*) | 11 = 3 Stellen |

Kann der erste Operand nicht alle Zeichen aufnehmen, so gehen die überzähligen Zeichen verloren.

Mnemonik zur Maskendefinition s. Abschnitt 4.20.

Aufbau des p 12-Bytes analog PACK-Befehl

Beispiel:

```
WERT:   12587Ø89-     (Gepacktes Format)
UNPK:   TPC*3, DRUCK (14), WERT (8)
DRUCK:  ***12.587,Ø89- (Ungepacktes Format)
```

| Namensfeld | Operationsfeld | Operanden- und Kommentarfeld | Translate |
|---|---|---|---|
| | TR | $adr_1[(l)]$ , $adr_2$ | TR |

**Befehlsformat:**

| 18 | L | adr1 | adr2 |
|---|---|---|---|

**Anzahl I-Bit:** 2

**Funktion:**

Codewandlung des ersten Operanden unter Verwendung des zweiten Operanden (Codetabelle)

**Bedingungscode:**

-

- Der Inhalt jedes Bytes des ersten Operanden wird zur Adresse adr2 addiert. Diese Summe ist die Adresse des Bytes aus der Codetabelle. Dabei entspricht der ersten Stelle der Codetabelle (adr2) die zu wandelnde Codierung H'ØØ', der zweiten Stelle die Codierung H'Ø1' usw. Das so adressierte Byte der Codetabelle überspeichert das zur Adreßbildung herangezogene Byte des ersten Operanden.
- Jede der 256 Bitkombinationen in einem Byte ist erlaubt.
- Beispiel:

```
TR      A1(5),A2
vor der Operation:
```

| | | | | | | |
|---|---|---|---|---|---|---|
| A1 | Ø5 | Ø7 | ØA | ØC | ØF | |
| | 5 | 7 | 1Ø | 12 | 15 | |
| A2 | | +1 | +4 | +5 | +6 | +7 | +8 | +9 | +10 | +11 | +12 | +13 | +14 | +15 |
| | ØØ | ØF | ØC | ØB | ØA | Ø9 | Ø8 | Ø7 | Ø6 | Ø5 | Ø4 | Ø3 | Ø2 | Ø1 |

nach der Operation:

| | | | | | |
|---|---|---|---|---|---|
| A1 | ØB | Ø9 | Ø6 | Ø4 | Ø1 |

## 8.8. Sprungbefehle

### 8.8.1. Unbedingter Sprung

| | | Branch |
|---|---|---|
| Namens-feld | Operations-feld / Operanden- und Kommentarfeld | B |
| B | adr | |
| Befehlsformat: ØD FØ adr | | Anzahl I-Bit: 1 |
| Funktion: Unbedingter Sprung | | |
| Bedingungscode: - | | |

- Der Befehl bewirkt ein Fortsetzen des Programms am Befehl, der bei adr beginnt.

## 8.8.2. Bedingte Sprünge

Das Programm wird nach der Adresse adr verzweigt, wenn die Verzweigungsbedingung erfüllt ist. Andernfalls wird das Programm mit dem nächsten Befehl fortgesetzt.

Die Verzweigungsbedingungen bleiben von den Sprungbefehlen unbeeinflußt.

### 8.8.2.1. Verzweigung in Abhängigkeit vom Bedingungscode

| Namensfeld | Operationsfeld | Operanden- und Kommentarfeld | Branch on condition |
|---|---|---|---|
| | BC<br>jmc | n, adr<br>adr | BC/ BH/ BL/ BE/ BNE/ BNL/ BNH/ BO/ BP/ BM/BZ/, BNZ/ BNM/ BNP/ BNO |

**Befehlsformat:**

| Ø | D | n | Ø | adr |
|---|---|---|---|---|

Anzahl I-Bit: 1

**Funktion:**

Bedingter Sprung in Abhängigkeit vom Bedingungscode

**Bedingungscode:**

-

| Verzweigungsbedingungen dezimal n | Sprung bei Bedingungscode | Erweiterter mnemonischer Op.-Code | | | | | |
|---|---|---|---|---|---|---|---|
| | | nach Vgl.Op. | | nach arithm.Op. | | nach TM-Bef. | |
| | | jmc | Bedeutg. | jmc | Bedeutg. | jmc | Wert d. mask. Bits |
| Ø | - | | - | | - | | - |
| 1 | 3 | - | - | BO | Überlauf | BO | alle 1 |
| 2 | 2 | BH | 1.Op. > 2. Op. | BP | positiv | - | - |
| 3 | - | - | - | - | - | - | - |
| 4 | 1 | BL | 1.Op. < 2. Op. | BM | negativ | BM | Ø u. 1-gem. |
| 5 | - | - | - | - | - | - | - |
| 6 | - | - | - | - | - | - | - |
| 7 | 1, 2, 3 | BNE | 1.Op. ≠ 2. Op. | BNZ | ≠Ø | BNZ | mind. eine 1 |
| 8 | Ø | BE | 1.Op. = 2. Op. | BZ | Ø | BZ | alle Ø |
| 9 | - | - | - | - | - | - | - |
| 10 | - | - | - | - | - | - | - |
| 11 | Ø, 2, 3 | BNL | 1.Op. ≥ 2. Op. | BNM | nicht neg. | BNM | alle Ø oder keine Ø |
| 12 | - | - | - | - | - | - | - |
| 13 | Ø, 1, 3 | BNH | 1.Op. ≤ 2. Op. | BNP | nicht pos. | - | - |
| 14 | Ø, 1, 2 | - | - | - | - | BNO | wenigstens eine Ø |
| 15 | | B | unbed. Spr. | B | unbed. Spr. | B | unbed. Spr. |

8.8.2.2.

## Verzweigung in Abhängigkeit vom Starttastencode

Nach den Befehlen ENT, ENTD, ENTP stehen die Starttastencodes im N-Register.

Nach den Befehlen ENTM, ENTU, ENTF stehen die Starttastencodes im A-Register.

Die Register A und N sind Systemregister und stehen dem Programmierer nicht zur Verfügung.

| Namensfeld | Operationsfeld | Operanden- und Kommentarfeld |
|---|---|---|
| | BCA<br>BCN | stc, adr |

Branch on condition of start-key A/N

BCA/BCN

**Befehlsformat:**

| Ø3 | | |
|---|---|---|
| Ø4 | stc | adr |

Anzahl I-Bit: 1

**Funktion:**

Programmverzweigung zur Adresse adr, falls der im Befehl angegebene Starttastencode mit der im Register A/N eingetragenen Codierung übereinstimmt

**Bedingungscode:**

-

Übersicht Starttastencodes stc

| stc | Tasten MPG | UBT | PRT | Code (hexa) | stc | Tasten MPG | UBT | PRT | Code (hexa) |
|---|---|---|---|---|---|---|---|---|---|
| ST/ET | S | | ET | ØØ | 32 | | | STT 32 | 2Ø |
| 1 | S 1 | PF 1 | STT 1 | Ø1 | 33 | | | STT 33 | 21 |
| 2 | S 2 | PF 2 | STT 2 | Ø2 | 34 | | | STT 34 | 22 |
| 3 | S 3 | PF 3 | STT 3 | Ø3 | 35 | | | STT 35 | 23 |
| 4 | S 4 | PF 4 | STT 4 | Ø4 | 36 | | | STT 36 | 24 |
| 5 | S 5 | PF 5 | STT 5 | Ø5 | 37 | | | | 25 |
| 6 | S 6 | PF 6 | STT 6 | Ø6 | 38 | | | | 26 |
| 7 | S 7 | PF 7 | STT 7 | Ø7 | 39 | | | | 27 |
| 8 | S 8 | PF 8 | STT 8 | Ø8 | 4Ø | | | | 28 |
| 9 | S 9 | PF 9 | STT 9 | Ø9 | 41 | | | | 29 |
| 1Ø | | PF 1Ø | STT 1Ø | Ø A | 42 | | | | 2 A |
| 11 | | PF 11 | STT 11 | Ø B | 43 | | | | 2 B |
| 12 | | PF 12 | STT 12 | Ø C | 44 | | | | 2 C |
| 13 | | | STT 13 | Ø D | 45 | | | | 2 D |
| 14 | | | STT 14 | Ø E | 46 | | | | 2 E |
| 15 | | | STT 15 | Ø F | 47 | | | | 2 F |
| 16 | | | STT 16 | 1Ø | 48 | | | | 3Ø |
| 17 | | | STT 17 | 11 | 49 | | | | 31 |
| 18 | | | STT 18 | 12 | 5Ø | | | | 32 |
| 19 | | | STT 19 | 13 | 51 | | | | 33 |
| 2Ø | | | STT 2Ø | 14 | 52 | | | | 34 |
| 21 | | | STT 21 | 15 | 53 | | | | 35 |
| 22 | | | STT 22 | 16 | 54 | | | | 36 |
| 23 | | | STT 23 | 17 | 55/CL | | | CL | 37 |
| 24 | | | STT 24 | 18 | 56/PA 3/ FV | | PA 3 | FV | 38 |
| 25 | | | STT 25 | 19 | 57/PA 2/ FB | | PA 2 | FB | 39 |
| 26 | | | STT 26 | 1 A | | | | | |
| 27 | | | STT 27 | 1 B | 58/PA 1/ SKIP | | PA 1 | SKIP | 3 A |
| 28 | | | STT 28 | 1 C | | | | | |
| 29 | | | STT 29 | 1 D | 59/SEND | | | SEND | 3 B |
| 30 | | | STT 3Ø | 1 E | 6Ø/CLEAR/ CA | | CLEAR | CA | 3 C |
| 31 | | | STT 31 | 1 F | 61/REC | | REC | REC | 3 D |
| | | | | | 62/ET 2/ CNCL | ET 2 | CNCL | ET 2 | 3 E |
| | | | | | 63/ET 1/ ENTER | ET 1 | ENTER | ET 1 | 3 F |

Anmerkung:

FV ≙ F FB ≙ F

8.8.2.3.

## Verzweigung in Abhängigkeit vom Selektor- bzw. Statuscode

Verzweigung ist bei erfüllter und nichterfüllter Bedingung möglich.

| Namens-feld | Operations- | Operanden- und Kommentarfeld |
|---|---|---|
| | BCK<br>BCKR<br>BCS | s 15, adr |

Branch on condition key/status

BCK/
BCKR/
BCS

**Befehlsformat:**

Ø6
Ø7
Ø5 | sc | adr |

**Anzahl I-Bit:**

1

**Funktion:**

Programmverzweigung zur Adresse adr, falls die durch s 15 angegebene Bedingung erfüllt ist

**Bedingungscode:**

-

Selektorcode s 15 (Assemblerschreibweise):

s 15 ::= { n / Nn }

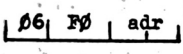
n - Verzweigung bei erfüllter Bedingung
Nn - Verzweigung bei nichterfüllter Bedingung

Selektorcode so (bei Programmierung im Maschinencode):

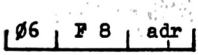
Selektorcode — i = Ø, Verzweigung bei erfüllter Bedingung
i = 1, Verzweigung bei nichterfüllter Bedingung

Beispiele:

1. BCK 15, MARKE1 — Sprung bei eingeschalteter Fehlerlampe




2. BCK N 15, MARKE1 — Sprung bei nichteingeschalteter Fehlerlampe




KR ist ein 2-Byte-Register, das die Belegung des K-Registers (zum Zeitpunkt der letzten Starttastenbetätigung) bei Beendigung des Tastatureingabebefehls übernimmt.

8.8.2.4.

## Verzweigung in Abhängigkeit von einem Zählwert

| Namensfeld | Operationsfeld | Operanden- und Kommentarfeld |
|---|---|---|
| | BCT | adr 1, adr 2 |

Branch on count

BCT

**Befehlsformat:**

| 3 D | adr 1 | adr 2 |
|---|---|---|

Anzahl I-Bit: 2

**Funktion:**

Verzweigen in Abhängigkeit von einem Zählwert

**Bedingungscode:**

-

Der erste Operand wird dekrementiert (Binärzahl).

Wird der erste Operand dabei Null, wird das Programm mit dem nach BCT folgenden Befehl fortgesetzt.

Ist der erste Operand ungleich Null, so wird das Programm nach der Adresse adr 2 verzweigt.

## 8.8.2.5. Verzweigung in Abhängigkeit von der Zeit

| Namensfeld | Operationsfeld | Operanden- und Kommentarfeld |
|---|---|---|
| | TIMER | adr 1, adr 2 |

Timer

TIMER

**Befehlsformat:**

| 3 F | adr 1 | adr 2 |
|---|---|---|

Anzahl I-Bit: 2

**Funktion:**

Verzweigung in Abhängigkeit von der Zeit

**Bedingungscode:**

-

Der erste Operand (adr 1) enthält die Sprungadresse.

Der zweite Operand (adr 2):

Keine indirekte Adressierung bei adr 2 (im Befehlsformat $I_2 = \emptyset$), dann ist adr 2 ein 2-Byte-Direktoperand (d 2) und gibt das Zeitintervall in Sekunden an

Indirekte Adressierung bei adr 2 (im Befehlsformat $I_2 = 1$ und im Assemblerschreibweise adr 2 in runde Klammern "( )" eingeschlossen), dann ist adr 2 eine Adresse, worauf das Zeitintervall in Sekunden steht.

Mit adr 2 = $\emptyset$ ist der Zeitgeber inaktiv (bzw. wird zurückgesetzt).

Ein neuer Befehl überschreibt vorhergehende Angaben.

Nach Abgabe des Befehls wird das Programm fortgesetzt. Die Verzweigung erfolgt an beliebiger Stelle nach Ablauf des Zeitwertes. Ein zum Zeitpunkt des Ablaufs des Zeitwertes anliegender E/A-Befehl wird unterbrochen.

Nach der Verzweigung befindet sich das Programm im Unterbrechungsstatus. Eine Fortsetzung an der unterbrochenen Stelle ist mit RET (analog Rückkehr aus dem Unterprogramm) oder Aufheben des Unterbrechungsstatus mit EXIT und Fortsetzen des Programms mit dem auf EXIT folgenden Befehl möglich.

Ein Unterbrechungsprogramm kann nicht erneut unterbrochen werden.

Ein in der 2. Programmebene durch TIMER vorgegebenes Zeitintervall wird bei Verlassen der 2. Programmebene nicht gelöscht. Somit wird die Adresse eines TIMER-Befehls, der vorher in der ersten Programmebene programmiert wurde, nach Ablauf dieses Zeitintervalls zur Unterbrechungsprogrammadresse.

8.8.2.6.
Verzweigung in Abhängigkeit des ON-Codes

| Namensfeld | Operationsfeld | Operanden- und Kommentarfeld | On (condition) |
|---|---|---|---|
| | ON | onc, adr | ON |

Befehlsformat:

| Ø8 | onc | adr |
|---|---|---|

Anzahl I-Bit: 1

Funktion:

Verzweigung zu adr, falls die im ON-Code (onc) spezifizierte Bedingung im weiteren Programmlauf auftritt

Bedingungscode:

-

Die unter adr angegebene Adresse wird zusammen mit der Bedingung onc in eine Tabelle eingetragen (Stack im SIEX).

Tritt eine für die Adresse angegebene Bedingung während der Abarbeitung des Programms auf, so wird in das Unterbrechungsprogramm (adr) verzweigt.
Durch den Befehl RET kann das unterbrochene Programm nach dem letzten abgearbeiteten Makrobefehl fortgesetzt werden. Durch den Befehl EXIT wird die Rückkehrverbindung aufgehoben (Stackkorrektur). Die Weiterarbeit erfolgt mit dem nach EXIT folgenden Befehl in der Programmebene, in der der ON-Befehl abgearbeitet wurde (ON-Befehle niederer Ebenen sind dann gelöscht).

Die erneute Programmierung eines ON-Befehls mit der gleichen Bedingung in der gleichen Programmebene löscht (überschreibt) die vorhergehende Verbindung.

Durch einen ON-Befehl mit adr = Ø wird eine evtl. bestehende Programmverbindung aufgehoben. In höheren Programmebenen bestehende Verbindungen werden dann wieder wirksam.
Soll eine ON-Verzweigung während des Programmlaufs generell verhindert werden, muß über adr eine Routine angesprungen werden, die ausschließlich aus dem Befehl RET besteht.

ON-Code:

| onc | Schreibweise |
|---|---|
| H'ØØ' ERROR (Systemfehler) | E |
| H'Ø1' Zero-Divide (Nulldivision) | Z |
| H'Ø2' Datenfehler | D |
| H'Ø3' POWER | P |

Hinweis:

1) Bei ERROR sollte ein organisierter Programmabbruch erfolgen.
2) POWER nur für Schalterterminal K 8924!
   Die verfügbare Zeit zwischen der Meldung und dem Abschalten des Systems über Betriebsspannungsabfall beträgt als Garantiezeit 10 ms.

## 8.8.2.7. Aufheben des Unterbrechungsstatus

| Namensfeld | Operationsfeld | Operanden- und Kommentarfeld |
|---|---|---|
| | EXIT | |

Exit

EXIT

**Befehlsformat:**

CB

Anzahl I-Bit:

0

**Funktion:**

Aufheben des Unterbrechungsstatus

**Bedingungscode:**

-

EXIT hebt den Unterbrechungsstatus auf und normalisiert den Stapelspeicher.

Es wird mit dem nach EXIT folgenden Befehl weitergearbeitet.

Programmlogisch wird die Ebene wie zum Zeitpunkt des zugehörigen Verbindungsbefehls eingenommen (TIMER, ON, ONF).

Der Befehl EXIT wird außerhalb eines Unterbrechungsprogramms überlaufen.

8.9.
UP-Arbeit

| Namensfeld | Operationsfeld | Operanden- und Kommentarfeld | Call |
|---|---|---|---|
| | CALL | adr | CALL |

**Befehlsformat:**

| Ø | B | adr |
|---|---|---|

Anzahl I-Bit: 1

**Funktion:**

Aufruf eines in Makrobefehlssprache geschrietenen Unterprogramms

**Bedingungscode:**

-

Dieser Befehl bewirkt den Aufruf eines Unterprogramms, das ebenso wie das rufende Programm in Makrobefehlssprache programmiert ist.

Bei Ausführung des Befehls wird zum Unterprogramm verzweigt und gleichzeitig die auf den CALL-Befehl folgende Adresse als Rückkehradresse in das rufende Programm gemerkt ( Organisation automatisch über Stack).

Parameterübergaben sind in den Befehlen PA und PL beschrieten.

| Namensfeld | Operationsfeld | Operanden- und Kommentarfeld |
|---|---|---|
| | CALC | adr |

Call CPU-subroutine

CALC

Befehlsformat:

| ØC | adr |
|---|---|

Anzahl I-Bit: 1

Funktion:

Aufruf eines in U 88Ø-Maschinensprache geschriebenen Unterprogramms

Bedingungscode:

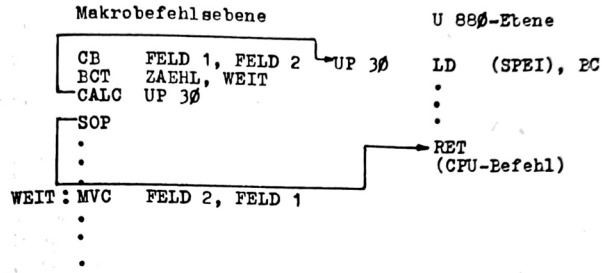
-

Dieser Befehl bewirkt den Aufruf eines UP, das in der U 88Ø-Maschinensprache programmiert ist.

Bei der Ausführung des Befehls wird der Interpreter verlassen (das UP nicht unter Steuerung des Interpreters ausgeführt) und zum UP verzweigt. Im Stack wird die Adresse nach dem CALC abgelegt.

Damit eine Rückkehr in den Interpreter und eine weitere Abarbeitung des rufenden Makroprogramms möglich wird, muß an der Rückkehrstelle im rufenden Programm der Befehl SOP codiert werden.

Beispiel:




| Namensfeld | Operationsfeld | Operanden- und Kommentarfeld |
|---|---|---|
| | CALP | adr |

Call Program

CALP

Befehlsformat:

CD | Ø9 | adr

Anzahl I-Bit:

$1_z$

Funktion:

Laden des unter adr angegebenen Programms aus der Phasenbibliothek und Starten des geladenen Programms

Bedingungscode:

-

Durch adr wird ein 5-Byte-Speicherbereich adressiert, der den Namen des zu ladenden Programms enthält.

Durch den Befehl CALP wird das Ladeprogramm des Betriebssystems aufgerufen. Das Programm, das den CALP-Befehl enthält, wird dabei beendet, sofern kein Fehler auftritt. Im Fehlerfall wird das Statusregister gesetzt.

Ladevorgang und Start sind völlig analog einem Programmaufruf durch Kommandoeingabe.

Dieser Befehl dient dem Aufruf eines Folgeprogramms für eine Anwenderprogrammfolge.

Die Datenübergabe wird nicht unterstützt und ist durch das Anwenderprogramm hauptspeicherresident oder über Datenträger selbst zu organisieren.

Overlaytechnik ist realisierbar, indem die Wurzelphase einem höheren Speicherbereich zugeordnet wird und relativ einen definierten Eintrittspunkt zum Ladepunkt der Programme besitzt.

| Namensfeld | Operationsfeld | Operanden- und Kommentarfeld |
|---|---|---|
| | RET | |

Return

RET

Befehlsformat:
C 1

Anzahl I-Bit:
ø

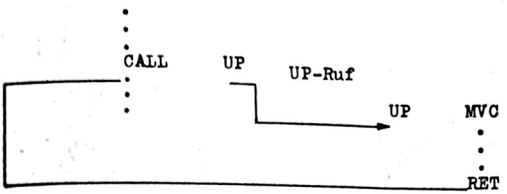
Funktion:
Rückkehr vom Unterprogramm in das rufende Programm

Bedingungscode:
-

Die Rückkehr vom Unterprogramm in das rufende Programm wird durch den Befehl RET realisiert.

Bei Ausführung wird zu der durch den CALL-Befehl gemerkten Rückkehradresse verzweigt.

Beispiel:




UP-Rücksprung

| Namensfeld | Operationsfeld | Operanden- und Kommentarfeld |
|---|---|---|
| | PA | 1, adr |

Parameter address

PA

Befehlsformat:

| Ø9 | 1 | adr |
|---|---|---|

Anzahl I-Bit:

1

Funktion:

Laden der Adresse der Parameterliste vom Haupt- in das Unterprogramm

Bedingungscode:

-

Dieser Befehl wird im Unterprogramm programmiert und ermöglicht eine Form der Übernahme von Parametern in ein Unterprogramm.

Im rufenden Programm müssen die zu übergebenden Parameter unmittelbar auf den UP-Aufruf CALL folgen.

1 gibt die Länge der Parameterblöcke in Bytes an.

Mit PA wird die Adresse des unmittelbar nach dem CALL stehenden Bytes ins UP übertragen und in adr, adr-1 abgespeichert.

PA erhöht außerdem die durch CALL in den Makro-Stack gebrachte Rückkehradresse um 1 (die Anzahl der Parameterbytes). Dadurch ist eine Rückkehr ins rufende Programm mit RET möglich.

| Namensfeld | Operationsfeld | Operanden- und Kommentarfeld |
|---|---|---|
| | PL | 1, adr |

Parameter list

PL

**Befehlsformat:**

| Ø A | 1 | adr |

Anzahl I-Bit: 1

**Funktion:**

Laden der Parameterliste vom Haupt- in das Unterprogramm

**Bedingungscode:**

-

Dieser Befehl wird im Unterprogramm programmiert und ermöglicht eine Form der Parameterübernahme in ein Unterprogramm.

Im rufenden Programm müssen die zu übergebenden Parameter unmittelbar auf den Unterprogrammaufruf CALL folgen.

Die durch 1 angegebene Anzahl von Bytes wird an die im 2. Operanden angegebene Adresse übertragen (vom rufenden in das aufgerufene Programm).

PL erhöht außerdem die durch CALL gemerkte Rückkehradresse um die Anzahl der übertragenen Parameterbytes in das rufende Programm. Hierdurch ist eine Rückkehr in das rufende Programm mit RET möglich.
Der Befehl PL wird in der Hauptebene überlaufen.

## 8.10. Sondersprünge

| Namensfeld | Operationsfeld | Operanden- und Kommentarfeld |
|---|---|---|
| | EOP | |

End of Program

EOP

**Befehlsformat:** C 2

**Anzahl I-Bit:** 0

**Funktion:**
Ende eines Programms

**Bedingungscode:**
-

Dieser Befehl kennzeichnet das Ende eines Programms.
Er bewirkt die Rückkehr in das Betriebssystem (Monitor).

| Namensfeld | Operationsfeld | Operanden- und Kommentarfeld | Start of Program |
|---|---|---|---|
| | SOP | | SOP |

| Befehlsformat: | Anzahl I-Bit: |
|---|---|
| RF | 0 |

Funktion:

Start eines Makroprogramms

Bedingungscode:

-

Der Befehl SOP kennzeichnet den Beginn eines Makroprogramms.

Er muß am Beginn (Eintrittspunkt) eines Makroprogramms programmiert werden (d.h. immer dann, wenn der Makrobefehlsinterpreter angesprungen wird, also auch nach einem CALC-Befehl).

SOP rettet keine CPU-Register, d.h. nach einer zeitweisen Einschaltung des Interpreters über SOP/EOI sind alle CPU-Register als undefiniert zu betrachten.

Entspricht BST 5 der CPU-Ebene.

Der Befehl SOP wird überlaufen, wenn der BS-Status bereits gesetzt ist, d.h. SOP folgt auf Makrobefehl.

| Namens-feld | Operations-feld | Operanden- und Kommentarfeld |
|---|---|---|
| | EOI | |

**End of interpreter**

EOI

**Befehlsformat:**

C Ø

**Anzahl I-Bit:**

Ø

**Funktion:**

Aufheben des Interpreterstatus

**Bedingungscode:**

-

Mit dem Befehl EOI wird der Interpreterstatus aufgehoben. Alle nach EOI folgenden Befehle werden als CPU-Befehle interpretiert.

Mit SOP kann in den Interpreterstatus zurückgelangt werden.

8.11.
Sonderbefehle

| Namensfeld | Operationsfeld | Operanden- und Kommentarfeld |
|---|---|---|
| | MVA<br>MVN | adr |

Move register A/N

MVA/
MVN

**Befehlsformat:**

68
69 | adr |

Anzahl I-Pit:
1

**Funktion:**

Transport der Starttastenregister A/N

**Bedingungscode:**

-

Der Inhalt der Starttastenregister A/N wird an die durch adr adressierte Stelle (1 Byte) transportiert.

| Namensfeld | Operationsfeld | Operanden- und Kommentarfeld |
|---|---|---|
| | MLF<br>MLS | adr |

**Modify length first/second**

MLF/MLS

**Befehlsformat:**

66
67 | adr |

Anzahl I-Bit: 1

**Funktion:**

**Modifikation** des zweiten/fünften Byte des Folgebefehls

**Bedingungscode:**

-

**In dem MLF/MLS folgenden Befehl wird das zweite/fünfte Byte durch das mit der Adresse adr adressierte Byte ersetzt.**

**Bei Programmen im ROM bzw. PROM sind die beiden Befehle nicht wirksam.**

**Beispiel:**

```
        .
        .
        .
MLF     A 1
A       A 2 (6), A 3 (3)
        .
        .
        .
<A 1>= 12
```

**Der Additionsbefehl wird ausgeführt mit dem ersten Operanden. Er hat zum Zeitpunkt der Ausführung eine Länge von 12 Bytes.**

| Namensfeld | Operationsfeld | Operanden- und Kommentarfeld |
|---|---|---|
| | NOP | |

No operation

NOP

Befehlsformat:

ØØ

Anzahl I-Bit:

Ø

Funktion:

Keine Operation (Leerbefehl)

Bedingungscode:

-

Es wird keine Operation ausgeführt.

| Namensfeld | Operationsfeld | Operanden- und Kommentarfeld |
|---|---|---|
| | OFF | |

Off

OFF

Befehlsformat:

CC

Anzahl I-Bit:

Ø

Funktion:

Ausschalten des Geräts

Bedingungscode:

-

Der Befehl bewirkt die Abschaltung des Geräts.

| Namensfeld | Operationsfeld | Operanden- und Kommentarfeld | Set/Reset Selector (group K,S) |
|---|---|---|---|
| | SETK<br>RESK<br>SETS<br>RESS | 1 15 | SETK/RESK<br>SETS/RESS |

**Befehlsformat:**

C 6
C 7
C 4
C 5 | 1 15

Anzahl I-Bit:

7

**Funktion:**

Setzen/Rücksetzen der Selektoren der Gruppe K/S

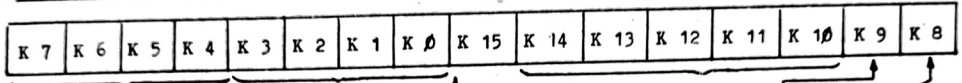
**Bedingungscode:**

-

**Alle in 1 15 ::=** $\left\{\begin{matrix} n \\ n, \quad 115 \end{matrix}\right\}$ **aufgeführten Elemente bewirken das Setzen/Rücksetzen der entsprechenden Selektoren (** $n$ **... 15).**

**Die nicht aufgeführten Elemente bewirken keine Änderung des aktuellen Selektorstandes.**

Selektorregister K




Nur über Progr. setz- u. löschbar

Über Progr. setz- u. löschbar; Anzeige über Leuchtdioden

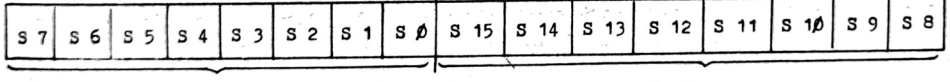
Durch Progr. u. System setzbar, durch Progr. u. CI-Taste (Fehlerlampe) löschbar

Bedienerkennung

Über Progr. setz- u. löschbar u. über Taste INS MODE setz- u. löschbar

Über Progr. löschbar, über Sicherungsbaugruppe setzbar (für Systemwertung)

Statusregister S




Durch E/A-Operationen eingestellt, außerdem durch Progr.

Für Programmierer frei verfügbar

| Namensfeld | Operationsfeld | Operanden- und Kommenterfeld | Guide |
|---|---|---|---|
| | GUIDE | adr | GUIDE |

**Befehlsformat:**

6 A | adr |

Anzahl I-Bit:

1

**Funktion:**

Benutzung des Befehls zur Bedienerführung

**Bedingungscode:**

-

Der im Befehl durch adr spezifizierte Operand (Länge 2 Byte) wird als 2 Zeichen interpretiert und in der Systemzeile des Bildschirms zur Anzeige gebracht.

| Namens-feld | Opera-tions-feld | Operanden- und Kommentarfeld |
|---|---|---|
| | SETD<br>SETC | adr |

Set data/clock

SETD/SETC

**Befehlsformat:**

CD Ø5
CD Ø4 | adr

Anzahl I-Bit:

$1_Z$

**Funktion:**

Eintragen der Zeit/des Datums in das Zeitregister/Datumregister des Betriebssystems

**Bedingungscode:**

-

In den durch adr adressierten 8 Byte ist das Datum/die Zeit in der Form dd.mm.yy (Tag.Monat. Jahr) / hh.mm.ss (Stunde. Minute.Sekunde) gespeichert.
(Anstatt '.' kann jedes beliebige Zeichen stehen)

Mit Einschalten der Anlage werden beide Register auf ØØ.ØØ.ØØ gestellt. Ab diesem Zeitpunkt wird die Zeit/das Datum gezählt.

| Namens-feld | Operations-feld | Operanden- und Kommentarfeld |
|---|---|---|
| | DATE<br>CLOCK | adr |

Date/Clock

DATE/
CLOCK

**Befehlsformat:**

CD Ø3
CD Ø2 adr

Anzahl I-Bit:

$1_Z$

**Funktion:**

Bereitstellen des aktuellen Datums/der Zeit in den durch adr adressierten 8 Byte

**Bedingungscode:**

-

In den durch adr adressierten 8 Byte wird das aktuelle Datum/die aktuelle Zeit in der durch SETD/SETC gespeicherten Form eingetragen.

Der Tag wird beim Übergang zur Zeitangabe ØØ.ØØ.ØØ um 1 erhöht. Die Weiterschaltung erfolgt bis dd = 99, mm und yy bleiben unverändert.

| Namens-feld | Operations-feld | Operanden- und Kommentarfeld |
|---|---|---|
| | ADRD<br>ADRE | adr |

Address of dialogne RAM/ end of RAM

# ADRD/ ADRE

**Befehlsformat:**

CD Ø1
CD ØØ adr

**Anzahl I-Bit:**

$1_Z$

**Funktion:**

Bereitstellen einer Systemadresse

**Bedingungscode:**

-

In der durch adr adressierten 2-Byte-Zeile werden folgende Adressen bereitgestellt:

ADRD: Anfangsadresse des Bildwiederholspeichers
ADRE: Endadresse des RAM (= Endadresse des Bildwiederholspeichers)

Ist kein Speicher zugeordnet, wird Ø eingetragen.

Die Differenz von ADRE und ADRD ergibt die Größe des Dialogbereichs.

| Namens-feld | Operations-feld | Operanden- und Kommentarfeld |
|---|---|---|
| | EX | adr 1, adr 2 |

Exchange

**Befehlsformat:**

| 3 E | adr 1 | adr 2 |
|---|---|---|

Anzahl I-Bit: 2

**Funktion:**

Befehlsaustausch mit Modifizierung

**Bedingungscode:**

Wird nur vom auszuführenden Befehl verändert

Der ab Adresse adr 2 stehende Befehl wird in das Befehlsregister geladen.

An der Adresse adr 1 steht ein Modifikationsbyte.

Das zweite Byte des im Befehlsregister stehenden Befehls wird mit dem Modifikationsbyte im Befehlsregister entsprechend dem "Logischen Oder" verknüpft, und der Befehl wird ausgeführt. Das Programm wird mit dem nach EX folgenden Befehl fortgesetzt.

Wenn der auszuführende Befehl ein Verzweigungsbefehl ist, wird das Programm bei erfüllter Sprungbedingung an der Ansprungadresse fortgesetzt.
(s. auch Befehl SETPS)

Wird durch EX ein Befehl angesprochen, der zu einem UP führt, so wird erst nach Rückkehr aus diesem bei dem nach EX folgenden Befehl fortgesetzt.

Werden EOI oder RET durch EX aufgerufen, so erfolgt nach Abarbeitung dieser Befehle keine Rückkehr zu dem Befehl, der dem EX-Befehl folgt (Wirkung wie bei Sprungbefehlen).

| Namensfeld | Operationsfeld | Operanden- und Kommentarfeld |
|---|---|---|
| | SETPS | d 1 |

Set program status

SETPS

Befehlsformat:

| CA | d 1 |

Anzahl I-Bit:

0

Funktion:

Setzen des Programmstatusbytes

Bedingungscode:

-

Es werden im Programmstatusbyte des Betriebssystems Anzeigen gesetzt bzw. gelöscht.

Das Programmstatusbyte ist nach Programmstart standardmäßig mit H'ØØ' eingestellt.

Aufbau des Operandenbytes:

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | Ø |
|---|---|---|---|---|---|---|---|

Zu einer Anzeige gehören immer 2 Bitstellen in d 1 (im Abstand von 4 Bit; also Bit (Ø und 4), (1 und 5),(2 und 6) und (3 und 7)).

Wenn im numerisch niederwertigen Bit eines solchen Paares eine 1 steht, wird die entsprechende Funktion eingestellt.

Wird zusätzlich im höherwertigen Bit eine 1 eingetragen, so wird die Funktion wieder gelöscht (Rückkehr in den Grundzustand).

Bit Ø,4: Freigabe der TYP-Funktion während der Abarbeitung des Anwenderprogramms

Bit 1,5: Anzeige der Systemzeile in Anwenderprogrammen

Bit 2,6: Unterdrückung der Fehleranzeige durch das Betriebssystem und der Bedienerquittung

Bit 3,7: Unterdrückung der Makrobefehlszähleränderung durch Befehle, die über EX aufgerufen wurden (keine Ausführung von Sprungbefehlen)

d.h.

| Bit Ø | 4 | |
|---|---|---|
| Ø | Ø | TYP verboten! |
| 1 | Ø | Freigabe TYP im Anwenderprogramm |
| 1 | 1 | TYP verboten! |

| Bit 1 | 5 | |
|---|---|---|
| Ø | Ø | Keine Anzeige der Systemzeile |
| 1 | Ø | Anzeige der Systemzeile im Anwenderprogramm einschließlich der voreingetasteten Ziffern bei numerischen Eingaben |
| 1 | 1 | Löschen der Anzeige |

| Bit 2 | 6 | |
|---|---|---|
| Ø | Ø | Fehleranzeige durch das Betriebssystem in der Systemzeile |
| 1 | Ø | Unterdrückung der Fehleranzeige/Fehlerbehandlung unter Regie des Anwenderprogramms |
| 1 | 1 | Rücknahme der Unterdrückung |

| Bit 3 | 7 | |
|---|---|---|
| Ø | Ø | Makrobefehlszähleränderung |
| 1 | Ø | Unterdrückung der Makrobefehlszähleränderung |
| 1 | 1 | Rücknahme der Unterdrückung |

8.12.
Eingabebefehle

Die Eingabebefehle werden unterschieden in

Direkteingabebefehle (ENT, ENTP, ENTM, ENTS)
Befehle zur Dialogarbeit (ENTF,ENTD,ENTU).

Bei den Befehlen ENT, ENTP, ENTM, ENTS erfolgt die Eingabe der Zeichen direkt in den Speicher, ohne Anzeige.

Bei den Dialogbefehlen ENTF, ENTD, ENTU wird der Anfang des Eingabebereichs durch die Kursorstellung festgelegt. Folglich muß vor der Dialogeingabe eine Kursorpositionierung durch POS-Befehl an den Anfang des Eingabebereichs erfolgen.

Zur Kursorarbeit bei Dialogeingaben:

Der Kursor erscheint erst mit anliegender Dialogeingabe auf dem Bildschirm.

Nach Ausführung der Eingabe in der letzten Eingabeposition des definierten Feldes rückt der Kursor in die nächste Zeile an die Anfangsposition (außerhalb des Feldes).
Nach Starttastenbetätigung verlischt der Kursor. Die Merkposition des Betriebssystems für die Kursorposition ist die erste Position außerhalb des definierten Feldes in der letzten Zeile.

Ausnahme:
Eingabezeilenlänge gleich Länge der Bildschirmzeile, dann ist die Merkposition die 1. Position der neuen Zeile unterhalb des Eingabefeldes.

Wird die Eingabe vor Erreichen des Feldendes abgebrochen, wird die Kursorposition der letzten Eingabestelle + 1 gemerkt.

Mit Anliegen der Befehle ENTF bzw. ENTD wird der adressierte Bereich gelöscht (NUL).

Dagegen erfolgt nach Eintritt in den Befehl ENTU keine Löschung des definierten Bereiches.

Die Dialogarbeit wird durch die Makrobefehle ENTD, ENTU, ENTF aktiviert und realisiert eine Direktverbindung von der Tastatur zu einem Dialogausgabegerät (Bildschirm, Drucker). Bei Anschluß des Druckers SD 1156 (nur für PRT 20) ist keine Dialogarbeit möglich.

Ist ein Bildschirmgerät vorhanden, so wird es durch das Startprogramm als Dialoggerät zugewiesen. Über das Kommando ASN kann eine abweichende Zuweisung durch Bedienerkommunikation erfolgen.

Bei der Dialogarbeit werden die Werte, über Tastatur eingegeben, sofort dem Dialoggerät übergeben. Die alphanumerischen Zeichen werden auf dem Dialoggerät dargestellt.

(Ausnahme: ENTD und Drucker als Dialoggerät. Hier erfolgt der Ausdruck erst nach Starttastenbetätigung.)

Es werden 2 Arten von Eingaben unterschieden:

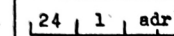
- Numerische Eingabe
  Zeichenvorrat beschränkt auf Ziffern Ø...9 und "-"
  Bereitstellung der eingegebenen Zeichen als Ziffer einschließlich Vorzeichen
  Rechtsbündige Einordnung der Zeichen
- Alphanumerische Eingabe
  Linksbündige Eingabe einer beliebigen Zeichenkette

Starttasten beenden die genannten Makrobefehle und damit die Dialogarbeit.
Die Funktionstasten für die Dialogarbeit sind in der Anleitung für den Systemprogrammierer, Teil I/4.2.4. beschrieben.

| Namens-feld | Operations-feld | Operanden- und Kommentarfeld |
|---|---|---|
| | ENT | adr [(l)] [,W] |

Enter

Befehlsformat:




Anzahl I-Bit:

1

Funktion:

Numerische Eingabe über die Tastatur mit Ablage im Speicher ab Adresse adr

Bedingungscode:

-

Die von der Tastatur eingegebenen Zeichen (Ziffern Ø,...,9,-) werden byteweise im Speicher ab Adresse adr in der Länge l (max. 16 Ziffernstellen und VZ-Stelle) rechtsbündig abgelegt.

adr = Ø ist nicht zugelassen.

Werden mehr als durch l-1 spezifizierte Zeichen eingetastet (Ausnahme: Korrektur vor Beendigung des Befehls), so erfolgt nach Bedienen einer Starttaste eine Fehlermeldung, die nach Betätigen der CI-Taste eine Neueingabe erfordert.

Der Befehl wird über Endetasten (Starttasten) beendet. Die Codierung der Starttasten wird in dem Systemregister N gespeichert. Werden weniger als durch l spezifizierte Zeichen eingegeben, so wird der Rest des Speicherbereichs mit Füllzeichen (NUL) aufgefüllt.

Mit der Starttastenbetätigung wird das Vorzeichen des numerischen Wertes festgelegt. Wenn vorher die Minustaste betätigt wurde, wird ein Minuszeichen (H'2 D') rechtsbündig (bei adr + 1 - 1) eingetragen, andernfalls ein Space (H'2Ø').

Die Tastatur arbeitet bei numerischen Eingaben gepuffert, d.h. alle eingetasteten Ziffern werden in einen Puffer eingeschrieben (Länge 16 Byte). Mit einem anliegenden ZIT-Befehl und nach betätigter Starttaste wird der Pufferinhalt in den adressierten Speicherbereich übertragen.

Die Anzeige auf dem Bildschirm ist möglich, wenn der adressierte Bereich (adr, 1) im Bildwiederholspeicher liegt.

Enter packed

ENTP

| Namensfeld | Operationsfeld | Operanden- und Kommentarfeld |
|---|---|---|
| | ENTP | n, adr [(k)] [,W] |

**Befehlsformat:**

| 21 | p 1Ø | adr |
|---|---|---|

Anzahl I-Bits: 1

**Funktion:**

Eingabe gepackter Dezimalzahlen über die numerische Tastatur mit Speicherung

**Bedingungscode:**

-

n - Eingabestellenzahl ($Ø \leq n \leq 15$)

Die über die Zehnertastatur in das Eingaberegister eingegebenen Informationen (Ziffern Ø, 1 ...9 und Vorzeichen) werden nach Quittierung mit einer Starttaste in gepackter Form (BCD) rechtsbündig ab der programmierten Adresse in den Hauptspeicher eingetragen.

Falsch eingegebene Ziffern können vor Startauslösung korrigiert werden.

Bei einer Überschreitung der programmierten Stellenanzahl (n) wird der Fehler angezeigt. Die Eingabe kann nach Drücken der Korrekturtaste wiederholt werden.

Aufbau des p 1Ø-Bytes:

| | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 |
|---|---|---|---|---|---|---|---|---|
| p 1Ø | ki | Ø | Ø | Ø | ←——n——→ | | | |

ki = Ø, bei Länge k =
1, bei Länge k =

| Namensfeld | Operationsfeld | Operanden- und Kommenterfeld |
|---|---|---|
| | ENTM | adr [(1)] [,W] |

Enter only in memory

ENTM

**Befehlsformat:**

| 25 | 1 | adr |
|---|---|---|

Anzahl I-Bit: 1

**Funktion:**

Eingabe von Zeichen über die Tastatur mit Speicherung

**Bedingungscode:**

-

**Funktion** analog ENTU-Befehl, aber ohne Ausgabe auf Dialoggerät

**adr = Ø** ist nicht zugelassen

**Die eingegebenen** Zeichen werden, beginnend ab adr, mit steigender Adreßfolge gespeichert. Eine Anzeige erfolgt nur, **wenn adr innerhalb** des Bildwiederholspeichers liegt.

**Erfolgt bei** der Zeicheneingabe die Überschreitung der **programmierten** Länge, blinkt die Fehlerlampe. Beseitigung **durch CI-Taste.**

Enter spezial

| Namensfeld | Operationsfeld | Operanden- und Kommentarfeld |
|---|---|---|
| | ENTS | adr[(1)][,W] |

ENTS

Befehlsformat:

| 22 | 1 | adr |
|---|---|---|

Anzahl I-Bit: 1

Funktion:
Eingabe von Zeichen über die Tastatur in den Speicher mit automatischen Abbruch bei erreichter Zeichenzahl

Bedingungscode:
-

- Funktion analog ENTN-Befehl; es erfolgt keine Ausgabe der Zeichen auf das Dialoggerät.
- adr = Ø ist nicht zugelassen!
- Eingabeschluß: 1. Starttasten
  2. Bei Erreichen des l-ten Zeichens A-Register: H'2E'
  3. Kursorsteuertasten, INS-LINE-Taste und Taste ERASE EOF

| Taste | A-Register |
|---|---|
| → | H'21' |
| INS-LINE | H'23' |
| ↑ | H'24' |
| ↓ | H'25' |
| ← | H'26' |
| → | H'27' |
| ↵ | H'2A' |
| ⇤ | H'2B' |
| ↰ | H'20' |
| ⇥ | H'2F' |
| ERASE EOF | H'28' |

(Bei vorzeitigen Abschluß ist der Feldrest mit H'ØØ' aufgefüllt)

Enter character formatted

| Namensfeld | Operationsfeld | Operanden- und Kommentarfeld |
|---|---|---|
| | ENTF | nz, adr [(l)][,w] |

ENTF

**Befehlsformat:**

| 23 | 1 | adr | nz |
|---|---|---|---|

Anzahl I-Bit: 1

**Funktion:**

Mehrzeilige Eingabe von Zeichen über die Tastatur

**Bedingungscode:**

-

Der Befehl ermöglicht die formatierte (mehrzeilige) Eingabe von Daten auf einen Speicherbereich mit der Anfangsadresse adr und der Länge l x nz.

Auf dem Bildschirm definieren l und nz die Kantenlängen eines Rechtecks mit:

l = Zeilenlänge
nz = Zeilenanzahl

Wertebereich: $1 \leq l, nz \leq 255$

Der linke Rand des Rechtecks ergibt sich aus der Horizontalposition des Dialoggeräts vor dem Befehl.

Die Eingabe wird über Endetasten beendet. Die Codierung der Endetasten wird in der Systemzelle A gespeichert. Werden weniger Zeichen eingegeben als durch **1 x nz** spezifiziert, wird der Rest des Speicherbereichs mit Füllzeichen (NUL) aufgefüllt. Das Dialoggerät bleibt auf der Position, an der die Endetaste betätigt wurde, stehen.

Nach Betätigen der NL-Taste wird eine Zeilenschaltung ausgeführt und eine Horizontaltabulation auf die Spalte vorgenommen, auf der sich das Dialoggerät unmittelbar vor Befehlsausführung befand (linker Rand).

Mit Eintritt in den Befehl wird das definierte Rechteck auf H'ØØ' gelöscht (auch im Speicher).

Ein Betätigen der Kursortasten führt den internen Speicherzeiger mit. Ein Verlassen des Eingabebereichs ist nicht möglich. Überschreiten der Eingabelänge durch Zeicheneingabe führt zu einer angezeigten Fehlerbedingung, die manuell beseitigt werden muß (CI-Taste).

| Namensfeld | Operationsfeld | Operanden- und Komentarfeld | Enter with display |
|---|---|---|---|
| | ENTD | adr 1 [(1)] ,adr 2 [,W] | ENTD |

**Befehlsformat:**

| 2Ø | 1 | adr 1 | adr 2 |
|---|---|---|---|

Anzahl I-Bit: 1

**Funktion:**

Numerische Eingabe über die Tastatur mit Anzeige auf dem Dialoggerät und Speicherung

**Bedingungscode:**

-

Die eingetasteten Ziffern werden auf das Dialoggerät ausgegeben (maximal 2Ø Zeichen).

Wird die Adresse adr 1 = Ø programmiert, so erfolgt keine Abspeicherung.

Mit Anliegen des Befehls wird die Interpunktionsmaske, die ab Adresse adr 2 abgespeichert ist, angezeigt.
Die Interpunktionsmaske bezieht sich auf Speicherbereich und Dialoggerät.

Aufbau der Interpunktionsmaske (s. PUT-Befehl)

Wenn adr 2 = Ø programmiert ist, erfolgt keine maskierte Anzeige. Die Ziffern werden in der eingetasteten Reihenfolge direkt angezeigt.

Die Ziffern werden an die adreßhöchste Ziffernstelle des Eingabebereichs ausgegeben (Kursor steht ebenfalls an adreßhöchster Stelle) bzw. erfolgt die Ausgabe ab aktueller

Druckwerksposition (Druck nach Beendigung des Eingabebefehls).

Nach Beendigung des Eingabebefehls steht der Kursor eine Stelle weiter rechts.

Mit jeder Tastenbetätigung wird die Ziffer im Dialogbereich angezeigt und bereits eingetastete Ziffern um 1 Stelle nach links entsprechend der Interpunktionsmaske verschoben.

Drucker als Dialoggerät:
Während der Druckausgabe ist ein Vereintasten verboten.

| Namensfeld | Operationsfeld | Operanden- und Kommentarfeld |
|---|---|---|
| | ENTU | nz, adr [(l)][,w] |

Enter character for updating

ENTU

**Befehlsformat:**

| 26 | 1 | adr | nz |
|---|---|---|---|

Anzahl I-Bit: 1

**Funktion:**

Ändern von Zeichen auf dem Dialoggerät über die Tastatur

**Bedingungscode:**

-

Der Befehl funktioniert wie der Befehl ENTF mit dem Unterschied, daß nach Eintritt in den Befehl kein Löschen des definierten Speicher- bzw. Bildschirmbereichs erfolgt.

Alle Zeichen im definierten Rechteck bzw. Speicherbereich bleiben erhalten, falls sie nicht durch Tastatureingabe überschrieben werden.

Es wird bei nicht vollständiger Kapazitätsausnutzung nicht mit H'ØØ' aufgefüllt.

Die Bildschirmsteuerzeichen (BSE, BSE, TAB) wirken nur bei diesem Befehl (s. auch Anleitung für Systemprogrammierer II.).

9.11.

## Ausgabebefehle für logische Geräte

Als logische Geräte g sind Drucker, Bildschirm, Lochband, Schreib-Leseeinheit und 1/2"-Magnetband zugelassen.

Wird der Befehl mit W programmiert (Intern wird dem Operationscode H'CF' vorgesetzt), dann wird nicht auf die Beendigung des Befehls gewartet.

Im weiteren Programmlauf muß der WAIT-Befehl folgen, sonst werden durch das Betriebssystem keine Fehler erkannt bzw. gemeldet.

Wird am WAIT-Befehl ein Fehler gemeldet, wird nach Betätigung der CI-Taste der dazugehörige Ausgabebefehl wiederholt (geräteabhängig).

| Namensfeld | Operationsfeld | Operanden- und Kommentarfeld |
|---|---|---|
| | PUT | g, adr 1[(l)] ,adr 2 [,W] |

Put

PUT

**Befehlsformat:**

28 | 1 | adr 1 | g | adr 2

Anzahl I-Bit: 1

**Funktion:**

Ausgabe numerischer Daten auf das logische Gerät g

**Bedingungscode:**

-

Die ab Adresse adr 1 stehenden l Bytes werden durchsucht und die enthaltenen Ziffern und das Vorzeichen auf das angegebene logische Gerät g ausgegeben. Dabei erfolgt eine Maskierung entsprechend der durch adr 2 adressierten Maske (der Länge 1).

Bei adr 2 = Ø erfolgt keine Maskierung.

Die Ausgabe erfolgt ab eingestellter Position (Kursor bzw. Druckkopf usw.) in der Länge l.

Nach der Ausgabe steht der Kursor bzw. Druckkopf usw. auf der nächsten freien Position.

PUT ohne WAIT ist nicht sinnvoll, wenn gepufferte Arbeitsweise des Geräts vorliegt.

Aufbau der Interpunktionsmaske:

Zeichenvorrat:

1. Zeichen mit Sonderbedeutung:
   - / + /b/ . / , / Ø /*
2. Ziffern: Ø . . . 9 (Ø wird wie 1 . . . 9 behandelt)
3. Sonstige Zeichen

Hinweis:
Die Zahl der Ziffernpositionen einschließlich Vorzeichen muß im Operand und Format übereinstimmen. Diesbezügliche Fehler werden nicht bemerkt.

Funktion:

| | |
|---|---|
| 1. - / + /b | kann an beliebigen Positionen eingeordnet werden.<br>- überschreibt diese Zeichen, sonst bleibt - / + /b erhalten. |
| 2. . / , | Interpunktion, Dezimalabtrennung; steht links von diesem Zeichen keine Ziffer oder kein *, wird das Zeichen in b gewandelt. |

Für die Dezimalabtrennung ist das gegebenenfalls durch Versetzen einer Ø im Format zu berücksichtigen.

3.x /Ø — Füllzeichen; werden durch Ziffern und Vorzeichen entsprechend der auszugebenden Zahl ersetzt. Sie bleiben in der Ausgabe als Sicherheitsstern bzw. Vornull erhalten.

4. Sonst. Zeichen — Sie werden bei der Ausgabe durch Ziffern überschreiben bzw. zu NUL gewandelt. Bei Dialogeingaben erscheinen die sonstigen Zeichen als Eingabeformat auf der Anzeigeeinrichtung und werden nach Abschluß der Eingabe gewandelt.

**Beispiele:**

| N = NUL Interpunktionsmaske | 1. Zahl | Ergebnis | 2. Zahl | Ergebnis |
|---|---|---|---|---|
| ////.// | 12345% | 1234.5% | 12345- | 1234.5- |
| ▪▪▪.// | N1279+ | ▪12.79+ | NN567,2 | ▪▪5.672 |
| ▪▪,▪▪▪ | NNN6% | ▪▪,▪6% | 83274 | 83,274 |
| //,/Ø,ØØ+ | NN89635 | NN689,63+ | NNNNN6% | NN6NØ,Ø6+ |
| //,//Ø.// | 123456- | 12,345.6- | NN12345 | NN123.45 |
| ØØ%ØØ% | 8727+ | 87527% | NN22- | ØØ522- |
| ▪▪.//,/% | NØØØ2- | ▪Ø.ØØ,2- | NNØ,15.3+ | ▪Ø.15,3 |
| Ø6//-//- | 3-35971+ | 3-59-71- | N+.2734 | Ø527-34- |

| Namens-feld | Operations-feld | Operanden- und Kommentarfeld |
|---|---|---|
| | PUTC | g, adr [(l)] [,W] |

Put character

PUTC

**Befehlsformat:**

| 2 | A | l | adr | g |
|---|---|---|---|---|

Anzahl I-Bit: 1

**Funktion:**

Zeichenausgabe aus dem Speicher auf ein logisches Gerät g

**Bedingungscode:**

-

Der Befehl bewirkt die Ausgabe von l Zeichen ab adr auf ein logisches Gerät g.

Die Ausgabe erfolgt ab eingestellter Position (des Kursors oder des Druckkopfes usw.).

Nach der Ausgabe steht der Kursor bzw. Druckkopf usw. auf der nächsten freien Position.

Sind innerhalb der l auszugebenden Zeichen die Codierungen H'ØA', H'ØC', H'ØD' und/oder H'Ø8' gespeichert, so werden die Funktionen, die den jeweiligen Codierungen entsprechen, vom Drucker bzw. Bildschirm während der Zeichenausgabe ausgeführt.

| Codierg. | Drucker mit CD | Drucker ohne CD | Bildschirm |
|---|---|---|---|
| H'ØA' | Zeilenschaltung | | Kursor neue Zeile |
| H'ØC' | Zeilenvorschub bis neue Seite | Papiervorschub vorwärts um die mit dem Schalter FF eingestellten Werte (4", 6", 8", 12") | Löschen ES |
| H'ØD' | Wagenrücklauf auf den linken Rand des jeweils eingestellten Formulars | Wagenrücklauf | Kursor an Zeilenanfeng |
| H'Ø8' | Rückschritt des Druckkopfes um eine Position | | Rückschritt des Kursors |

Ist die Lochbandeinheit K 6200 als logisches Gerät g angeschlossen, so erfolgt die Ausgabe von (-Zeichen über den LB-Stenzer.

Analog den Druckern SD 1152 und SD 1156 werden bei Anschluß des Druckers SD 1157 folgende Steuerzeichen erkannt und ausgeführt:

H'ØA' LF Zeilenschaltung

H'ØC' FF Zeilenvorschub bis neue Seite bei Arbeit mit CD; Papiervorschub vorwärts um die mit dem Schalter FF eingestellten Werte (4", 6", 8", 12") bei Arbeit ohne CD;

H'ØD' CR Wagenrücklauf
H'Ø8' BS Rückschritt des Druckkopfes von 1 Position

Da beim Drucker SD 1157 die Möglichkeit der Umschaltung auf einen anderen Zeichenbereich besteht, können die folgenden Steuerzeichen zusätzlich ausgegeben werden:

H'Ø ' SO Umschaltung auf den zweiten internen Zeichenbereich (z.B. kyrillisch)
H'ØF' SI Umschaltung auf den ersten internen Zeichenbereich

Bei den Druckern SD 1152 und SD 1156 werden diese Steuerzeichen zu Space (H'2Ø') gewendelt.

Einschränkungen für die Arbeit mit Breitdruck (bei Betriebssystemversion Ø3 - gilt für die Befehle PUTC, PUT, PUTF, PUTP, ENTD):
Es kann nur bedingt mit Breitdruck gearbeitet werden. Da die Zeichen doppelte Breite haben, stimmen die tatsächliche Druckposition und die intern gespeicherte Druck-Istposition nach dem Druck nicht mehr überein. Das Zeilenende wird nicht erkannt, wenn sich der Druckkopf am rechten Rand befindet.

Bevor der nächste Druck mit Normal- oder Schrägdruckschrift erfolgt, sollte stets ein POS-Befehl mit Horizontal absolut programmiert werden, damit die Druck-Istposition mit der Druckkopfposition wieder übereinstimmt.

Ist als logisches Gerät g die Schreib-Leseeinheit (SLE) angeschlossen, dann bewirkt der Befehl PUTC das Aufzeichnen der ab adr im ISO-7-Bit-Code bereitgestellten Daten auf die Magnetkarte. Die Daten werden durch den Modul automatisch, je nach Spur, in den 4- bzw. 6-Bit-Code gewandelt.
.l gibt an, auf welche Spuren auszugeben ist. (Siehe GET-Befehl.)
.Trenn- und Sonderzeichen müssen wie Daten im Ausgabebereich stehen.
.Start-, Stop- und LRC-Zeichen werden vom Modul erzeugt und sind vom Anwender nicht in den Ausgabebereich einzutragen.

. Nach der Schreiboperation wird die Magnetkarte automatisch zurückgefahren und ein Kontrollesen durchgeführt. Die Karte befindet sich ausschließlich in der Leseendposition.
. Wurde Schreibfehler erkannt, kann mit CTRL fetc=BWR und erneuten PUTC der Vorgang wiederholt werden.

- Programmierung für Anschluß 1/2"-Magnetbands siehe Sonderheft!

| Namensfeld | Operationsfeld | Operanden- und Kommentarfeld |
|---|---|---|
| | PUTP | g, m, n, adr [(k)][,W] |

Put packed

PUTP

**Befehlsformat:**

29 | p 10 | adr | g | m |

Anzahl I-Bit: 1

**Funktion:**

Ausgabe gepackter Dezimalzahlen auf das logische Gerät g mit Maskenaufbereitung

**Bedingungscode:**

-

Im Hauptspeicher stehende numerische (gepackte) Daten können mit diesem Befehl direkt (ohne vorheriges Entpacken) auf das logische Gerät g ausgegeben werden.

Die Zeichenaufbereitung erfolgt entsprechend der programmierten Maske.

Aufbau des Maskenbytes m analog UNPK-Befehl!

Die Informationen zur Interpunktion und zur Dezimalstellenabtrennung können weggelassen werden.

Standardmäßig wird dann eingetragen:

NI = ohne Interpunktion
K = Dezimalstellenabtrennung durch Komma

Beispiel:

C'Z2'

Keine Interpunktion
Dezimalstellenabtrennung durch Komma
Führende Stellen als Null
2 Dezimalstellen

C'TPS1'

Tausenderinterpunktion (Punkt)
Dezimalstellenabtrennung (Komma)
Führende Stellen als SPACE
1 Dezimalstelle

Die Informationen über Behandlung führender Stellen und Anzahl Dezimalstellen müssen immer geschrieben werden.

Mit Ausführung des Befehls PUTP bleiben die Daten in unveränderter Form (gepackt) im Hauptspeicher erhalten.

Stehen im adressierten Hauptspeicherbereich mehr Ziffern als mit der Stellenanzahl programmiert wurden, dann wird die Ausgabe nicht ausgeführt und der Fehler angezeigt.

Aufbau des p 10-Bytes analog ENTP-Befehl

| Namens-feld | Operations-feld | Operanden- und Kommentarfeld |
|---|---|---|
| | PUTF | g, nz, adr [(l)] [,W] |

Put formatted

PUTF

**Befehlsformat:**

| 2 B | l | adr | g | nz |
|---|---|---|---|---|

Anzahl I-Bit: 1

**Funktion:**

Mehrzeilige Ausgabe auf ein logisches Gerät

**Bedingungscode:**

-

Der Befehl bewirkt die formatierte (mehrzeilige) Ausgabe von l x nz Zeichen ab adr auf ein logisches Gerät g ab aktueller Position.

l ist die Anzahl der Zeichen, die maximal in einer Zeile ausgegeben werden sollen ($1 \leq l \leq 255$).

nz ist die Anzahl der Zeilen, die ausgegeben werden sollen ($1 \leq nz \leq 255$).

Die Ausgabe erfolgt in der Länge l, danach wird jeweils eine Zeilenschaltung ausgeführt, anschließend auf die Ausgangsposition zu Beginn des Befehls zurückgesetzt und die Ausgabe bis zum Erreichen nz fortgesetzt.

Nach der Ausgabe steht der Kursor (bzw. Druckkopf) hinter dem zuletzt ausgegebenen Zeichen (auf der nächsten freien Position).

| Namensfeld | Operationsfeld | Operanden- und Kommentarfeld | Get |
|---|---|---|---|
| | GET | g, adr [(l)] [,W] | GET |

**Befehlsformat:**

| 2 D | 1 | adr | g |
|---|---|---|---|

Anzahl I-Bit: 1

**Funktion:**

Eingabe von einem logischen Gerät g

**Bedingungscode:**

-

Der Befehl bewirkt die Eingabe in den Speicher von einem logischen Gerät g in der Länge l ab adr.

Als logische Geräte g sind hier nur Bildschirm, Lochband und Magnetkontokarte zugelassen.

Beim Anschluß der Lochbandeinheit K 6200 werden 1 Zeichen über den LB-Leser eingelesen, wobei NUL und DEL überlesen werden, wenn vorher CTRL mit fctc::=NND programmiert wurde. Sollen alle Zeichen einschließlich NUL- und DEL-Stanzung in den E/A-Bereich gelesen werden, muß vor dem GET-Befehl der CTRL mit fctc::=RND programmiert werden.

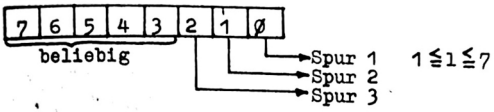
Beim Lesen von Lochbändern ist durch das Anwenderprogramm selbständig auf eine vom Anwender definierte Endemarke (Datenende Lochband) zu prüfen.
(Prüfung erfolgt im Eingabefeld des GET-Befehls.)
Beachte: Für LB-Arbeit sind die fertigen UP's zu nutzen!

Ist als logisches Gerät g die Schreib-Lese-Einheit (SLE), die automatische Leseeinheit (ALE) oder die Handleseeinheit (HLE) angeschlossen, dann bewirkt der Befehl das fortlaufende

Einlesen der Daten der zu lesenden Spuren der Magnetkarte in den Speicher ab adr. Die 4- bzw. 5-Bit-Code-Daten werden vom Modul in den ISO-7-Bit-Code gewandelt.

. 1 gibt an, welche Spuren gelesen werden sollen.

Aufbau des 1-Bytes:




Bit = Ø Spur wird nicht gelesen/geschrieben
1 Spur wird gelesen/geschrieben

. Werden alle 3 Spuren gelesen, stehen die Daten in der Folge der Spuren 1,2,3 im Eingabebereich ab adr. Wird Spur 1 nicht gelesen, so stehen ab adr die Daten von Spur 2 und danach von Spur 3.

. Start-, Stop- und LRC-Zeichen werden nicht in den Speicherbereich gelesen.

. Trenn- und Sonderzeichen innerhalb der Daten werden wie Daten behandelt.

. Zur Erhöhung der Lesesicherheit wird auf festes Längenformat kontrolliert.

. Nach der Leseoperation befindet sich die Magnetkarte in der Leseendposition.

. Nach Lesefehlern kann mittels CTRL mit fctc:=BWR und erneutem GET der Lesevorgang wiederholt werden.

. Kann auf dem Magnetstreifen kein gültiges Startzeichen erkannt werden, endet die Operation in der Leseendposition, ohne das Daten eingelesen wurden. Es erfolgt eine entsprechende Fehlermeldung.
Gelesen wird, bis ein gültiges Stop-Zeichen erkannt wird. Ist kein Stop-Zeichen gültig, wird bis Leseendeposition gelesen und Fehler gemeldet.

Bei Anschluß des 1/2"-Magnetbandes Programmierhinweise im Sonderheft beachten!

## 8.14.
## Befehle zur Steuerung peripherer Geräte

| Namensfeld | Operationsfeld | Operanden- und Kommentarfeld |
|---|---|---|
| | POS | g [(c)][,pperlist][,W] |

Position: POS

**Befehlsformat:**

2 C | par | $H^{d2}$ | V | g

Anzahl I-Bit: Ø

**Funktion:**

Positionierung und Steuerung von Geräten mit optisch lesbarer Datenausgabe

**Bedingungscode:**

-

**Positionierung auf einem Ausgabegerät**

Das nach dem Operationscode folgende Parameterbyte bestimmt die Interpretation der folgenden zwei Byte.

Aufbau des Parameterbytes:

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | Ø |
|---|---|---|---|---|---|---|---|
| Ø = Direktwert H, V<br>1 = Adresse, unter der H, V abgespeichert sind | horizontal | | vertikal | | Ø - Grundstellung (R)<br>1 - Umkehrung | Teilfunktionsauswahl | |

**horizontal:** + ØØ relativ vorwärts
- Ø1 relativ rückwärts
= 1Ø absolut
FF 11 Form feed (neue Seite), H absolut, V absolut und relativ
Drucker: Zeilenvorschub bis neue Seite
Bildschirm: Löschen des Bildschirms
bei fehlender Steuertabelle: H absolut
Konteneinzug KE 1161 auf SD 1152: Kartenaustrieb. Beim SD 1152 können mit Drehschalter 4 verschiedene Längen am Drucker eingestellt werden.
Mit FF wird dann die eingestellte Länge ausgegeben.

**vertikal:** + ØØ relativ vorwärts
- Ø1 relativ rückwärts (bei KE 1161 nicht möglich, es wird Positionsfehler gemeldet)
= 1Ø absolut
VT 11 Zeilenvorschub bis Stoppunkt (lt.CD-Tab.)
Wenn keine CD vorhanden, dann "=" und "VT" ohne Wirkung
Bei BAB kein VT möglich! (Codierung 11=1Ø).

**Umkehrung (Reverse R):**

**Drucker:** Keine Angabe Ø - Einstellen des über CTRL definierten Farbgrundzustandes beim SD 1152 bzw. Einstellen des Grundzustandes der Druckschriftart beim SD 1157.

R 1 - Einstellen des über CTRL definierten Umkehrzustandes des Farbbandes beim SD 1152 bzw. der Druckschriftart beim SD 1157.

**Anmerkung:** SIEX stellt standardmäßig Schwarzdruck (beim SD 1152) bzw. Normaldruck (beim SD 1157) ein. Rotdruck beim SD 1152 entspricht Schrägdruck bei SD 1157!

Teilfunktionsauswahl g (o):

Drucker:

| g (o) | Bit 1 | Bit 0 | Teilfunktion |
|---|---|---|---|
| g oder g (1) | 0 | 0 | Leporello 1 bzw. Journal links |
| g (2) | 0 | 1 | Leporello 2 bzw. Journal rechts |
| g (3) | 1 | 0 | KE oder anderes Aufsatzgerät |

Assemblerschreibweise:

pparlist ::= { ppar | pparlist,ppar } (Positionierparameterliste)

ppar ::= { H { + | - | = } g | { V { + | - | = } g | VT } | I = adr | FF | R }

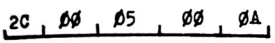
Mit dem TOS-Befehl ist eine Simultanarbeit möglich, wenn dem Operationscode die Codierung H'CF' vorgesetzt wird. (Bei Programmierung in Assemblersprache: Angabe von "W" im Operandenfeld.) Im weiteren Programmlauf muß der WAIT-Befehl mit der logischen Adresse des Druckers folgen. Sonst erfolgt keine Fehlererkennung bzw. Meldung über das Betriebssystem.

**Standardfall:**

Parameterbyte = Ø
relativ vorwärts
Leporello 1 bzw. Journal links
Schwarzdruck

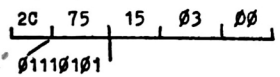
**Beispiele:**

1. POS 1Ø, H + 5




Es werden auf dem Ausgabegerät g =1Ø fünf Horizontalschritte nach rechts ausgeführt.

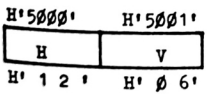
2. POS Ø(2), H = 21, V = 3, FF, R




Es wird auf dem Ausgabegerät g = Ø folgendes ausgeführt:
Lep. 2 bzw. Journal rechts angewählt
Vorschub auf eine neue Seite
Einstellen der Position 21 horizontal ab linkem Rand (CD) und 3 vertikal
Umschaltung auf Rotdruck

3. POS 11(3), H - Ø, I = H'5ØØØ'

Ausgabegerät g = 11, Kontokarteneinzug angewählt
Positionierung horizontal relativ rückwärts und vertikal relativ vorwärts mit dem Wert, der auf der 2-Byte-Zeile ab der Adresse H'5ØØØ' steht




Es werden 18 Horizontalschritte rückwärts und 6 Vertikalschritte vorwärts ausgeführt.
Achtung: Vertikal relativ rückwärts bei KE verboten!

| Namensfeld | Operationsfeld | Operanden- und Kommentarfeld |
|---|---|---|
| | CD | g, adr |

Control dictoionery

CD

**Befehlsformat:**

| 27 | par | adr | g |

**Anzahl I-Bit:**

1

**Funktion:**

Definition einer Steuertabelle

**Bedingungscode:**

-

**Aufbau:**

1. 1 Byte; Länge der Teiltabelle 1 dual in Byte (Teilfunktion 1)
2. 1 Byte; Zeilenzahl je Seite
3. 1 Byte; Position des linken Randes
4. 1 Byte; logisch letzte Zeile
5. m Bytes; Stoppunkte in aufsteigender Reihenfolge

analog Teiltabelle 2 (Teilfunktion 2) und Teiltabelle 3 (Teilfunktion 3).

Die Definition der Steuertabelle ist mit dem Rücksetzen aller Zeilenzähler auf H'ØØ' verbunden.

Das Parameterbyte (zweites Befehlsbyte) wird mit H'ØØ' codiert.

Wird mit g die logische Gerätesdresse der Tastatur bestimmt, dann bezeichnet adr eine 2-Byte-Zeile, in der bei jedem Tastatureingabebefehl mit Ablage im Speicher (adr ≠ Ø) die Anzahl der eingetasteten Zeichen abgespeichert wird.

Bei Anschluß des 1/2"-Magnetbandes Programmierhinweise im Sonderheft beachten!

| Namensfeld | Operationsfeld | Operanden- und Kommentarfeld |
|---|---|---|
| | CTRL | adr, fctc [,W] |

Control

CTRL

**Befehlsformat:**

| 2 E | fctc | adr |
|---|---|---|

Anzahl I-Bit: 1

**Funktion:**

Steuerung peripherer Geräte

**Bedingungscode:**

-

Entsprechend der Anmerkung bei den Funktionscodes bezeichnet adr die Anfangsadresse des Dateidefinitionsblocks oder die logische Adresse des Geräts (Im letzteren Fall darf adr nur im Wertebereich von Ø ... 255 liegen).

**Funktionscodes:**

1. KMBG K 5200:

| Codierung | fctc | Funktion | |
|---|---|---|---|
| H'ØØ' | TRP | Entriegeln | adr = Log. Geräteadresse |
| H'Ø2' | FOR | Vorspulen bis Klarsichtband und Entriegeln | |
| H'Ø3' | REW | Rückspulen bis Klarsichtband und Entriegeln | |
| H'Ø4' | FSF | Vorsetzen über Bandmarke mit physischem Eröffnen, falls entriegelt | |
| H'Ø5' | BSF | Rücksetzen vor Bandmarke | |
| H'3Ø' | WTM | Schreiben Bandmarke mit physischem Eröffnen, falls entriegelt | |
| H'C4' | BSR | Einen Block in der Datei rücksetzen | adr = DDB-Adr. |

2. Drucker SD 1152

| Codierung | fctc | Funktion | |
|---|---|---|---|
| H'ØØ' | BEC | Farbbandgrundstellung schwarz | adr = Log. Geräte-adresse |
| H'Ø8' | RBC | Farbbandgrundstellung rot | |

3. Drucker SD 1156 (spezielle Hardware + Schneideinrichtung nur für PRT 20

| Codierung | fctc | Funktion | |
|---|---|---|---|
| H'ØØ' | CUT | Schneiden | adr = Log. Geräte-adresse |
| H'Ø1' | FLP | Zeilenschaltung 1/12" vorwärts Leporello 1 | |
| H'Ø2 | BLP | Zeilenschaltung 1/12" rückwärts Leporello 1 | |
| H'11' | FL2 | Zeilenschaltung 1/12" vorwärts Leporello 2 | |
| H'12' | BL2 | Zeilenschaltung 1/12" rückwärts Leporello 2 | |

4. Bildschirm (BAB)

| Codierung | fctc | Funktion | |
|---|---|---|---|
| H'ØØ' | KPN | Kursor ruhend | adr = Log. Geräte-adresse |
| H'Ø1' | KPI | Kursor blinkend | |
| H'Ø2' | SDC | 48Ø Zeichen | |
| H'Ø3' | BDC | 192Ø Zeichen | |

Erläuterung:

| | | | |
|---|---|---|---|
| TPR | trip | CUT | cut |
| FOR | forward | FLP | forward line feed Lep1 |
| REW | rewind | BLP | backward line feed Lep1 |
| FSP | forward space file | FL2 | forward line feed Lep2 |
| BSP | backspace file | BL2 | backward line feed Lep2 |
| WTM | write tape mark | KPN | kursor presentation normally |
| BSR | back space record | KPI | kursor presentation intermittent |
| BBC | black basic collour | SDC | small display-capacity |
| RBC | red basic collour | BDC | big display-capacity |

## 5. Lochbandeinheit K 6200

| Codierung | fctc | Funktion | |
|---|---|---|---|
| H'ØØ' | BSS | Rücksetzen Stanzer um 1 Stanzposition | adr = Log. Geräteadresse |
| H'Ø1' | DEL | Stanzen des Zeichens "Delete" (Stanzen aller 8 Spuren) | |
| H'Ø2' | RND | Lesen einschl. NUL und DEL bei GET-Befehl | |
| H'Ø3' | NND | Lesen aller Zeichen außer NUL und DEL bei GET-Befehl | |

Erläuterung:

BSS back space 1 Step
DEL delete
RND read nul/del
NND read no nul/del

## 6. Drucker SD 1157

| Codierung | fctc | Funktion | |
|---|---|---|---|
| H'ØØ' | BBC | Grundstellung ist Normaldruck | adr = Log. Geräteadresse |
| H'Ø8' | RBC | Grundstellung ist Schrägdruck | |
| H'Ø6' | WON | Breitdruck einschalten | |
| H'ØC' | WOF | Breitdruck ausschalten | |
| H'Ø1' | FLF | Zeilenschaltung 1/12" vorwärts Lep1 | |
| H'Ø2' | BLF | Zeilenschaltung 1/12" rückwärts Lep1 | |
| H'11' | FL2 | Zeilenschaltung 1/12" vorwärts Lep2 | |
| H'12' | BL2 | Zeilenschaltung 1/12" rückwärts Lep2 | |

Erläuterung:

WON wide text on
WOF wide text off
BBC, RBC, FLF, BLF, FL2, BL2 } analog SD 1152

7. SLE, ALE, MLE, für Magnetkarten

| Codierung | fctc | Funktion |
|---|---|---|
| H'ØØ' | TRP | Abbruch eines nicht ausgeführten PUTC-/GET-Befehls x) |
| H'Ø1' | PUO | Ausstoßen Magnetkarte |
| H'Ø2' | BWR | zurück zur Schreib-Lese-Anfangsposition |
| H'Ø3' | SPO | Eintragung der Kartenposition in das S-Register |
| H'Ø4' | CFC | Magnetkarte beschlagnahmen |
| H'Ø5' | STS | Einschalten der akustischen Entnahmeaufforderung |
| H'Ø8' | GBO | Verriegeln des Eingabetores |

Erläuterung:

| | |
|---|---|
| TRP | trip |
| PUO | push out |
| BWR | back to write-read-position |
| SPO | send position |
| CFC | confiscate |
| STS | set take away signal |
| GBO | gate bolt |

Beachte: CTRL mit fctc:=SPO darf nicht mit "W" programmiert werden! Es muß in jedem Fall auf Ausführung des Befehls gewartet werden (d.h. keine Simultanarbeit möglich!).

8. 1/2"-Magnetband

Siehe Sonderheft!

x) Bedingung: Karte befindet sich noch nicht im Gerät bzw. erst in Einzugsposition.

Ablauf: Kartenantrieb wird gestoppt, alle Kartenpositionen bleiben erhalten, logische Rücknahme des PUTC-/GET-Befehls.

## 8.15. Befehle zur Dateiarbeit

| Namensfeld | Operationsfeld | Operanden- und Kommentarfeld |
|---|---|---|
| | OPEN | adr |

Open "data set"

OPEN

**Befehlsformat:**

| 6 | 3 | adr |
|---|---|---|

Anzahl I-Bits: 1

**Funktion:**

Eröffnen einer Datei

**Bedingungscode:**

-

Der Befehl muß vor dem ersten Zugriff auf die Datei ausgeführt werden. Es wird im System vermerkt, daß die Datei eröffnet ist. Die Operation läuft unter Einbeziehung des DDB (statischer Dateidefinitionsblock) ab, der durch adr im Befehl verankert ist. Der DCB wird initialisiert. Die aktuelle Satzadresse wird auf H'ØØØØ' gesetzt.

Der Befehl "OPEN" wird bei Anwendung auf eine bereits eröffnete Datei ignoriert.

Aus der Angabe der logischen Adresse wird das zugeordnete physische Gerät ermittelt und die Kennsatzprüfung (wenn Kennsatz vorhanden) vorgenommen. Bei Nichtübereinstimmung wird der Datenträger zurückgewiesen.

Sind im zugehörigen DDB die ERR- bzw. EOF-Adresse ungleich Ø angegeben, so erfolgt die gleiche Behandlung wie bei Abarbeitung eines äquivalenten OMF-Befehls.

| Namensfeld | Operationsfeld | Operanden- und Kommentarfeld |
|---|---|---|
| | CLOSE | adr |

Close "data set"

CLOSE

**Befehlsformat:**

| 6 | C | adr |
|---|---|---|

**Anzahl I-Bit:** 1

**Funktion:**

Abschließen einer Datei

**Bedingungscode:**

-

Der Befehl muß nach dem letzten Zugriff zur Datei ausgeführt werden. Es wird im System vermerkt, daß die Datei abgeschlossen ist. Die Operation läuft unter Einteziehung des DDB (statischer Dateidefinitionsblock) ab, der durch adr im Befehl verankert ist.

Die Dateiendebehandlung wird auf dem Datenträger vorgenommen, und der dynamische Dateidefinitionsblock wird freigegeben.

**Close-Option:**

COPY (Codierung ØØ): Datei unverändert
SAVE (Codierung 1Ø): Setzen des Schreibschutzanzeigers
ERASE (Codierung Ø1): Dateikennsatz wird ungültig
DEL (Codierung 11): Löschen der Datei (aber Dateikennsatz bleibt erhalten)

Der Befehl CLOSE wird ignoriert, wenn er auf eine bereits geschlossene Datei angewendet wird.

| Namensfeld | Operationsfeld | Operanden- und Kommentarfeld | Read sequentially<br>Write sequentially<br>READ/WRITE |
|---|---|---|---|
| | READ<br>WRITE | adr 1, adr 2, pos [,pos] | |

**Befehlsformat:**

40
44 | s 2 | adr 1 | adr 2 |

Anzahl I-Bit:
$2_Z$

**Funktion:**

Lesen/Schreiben eines Satzes von/auf den Datenträger

**Bedingungscode:**

-

Die Operation läuft unter Einbeziehung des Dateidefinitionsblocks DDB ab, dessen Anfangsadresse durch adr 1 im Befehl angegeben wird.

Der adressierte Satz wird in/aus den durch adr 2 im Befehl gekennzeichneten Bereich gelesen/geschrieben. Die Adressierung erfolgt durch "pos" und die aktuelle Satzadresse.

**Als Satzpositionierung (pos) sind zugelassen:**

**N (next)** (Codierung ØØØ): Positionierung auf den nächsten ((n + 1)-ten) Satz des Datenbestandes

**B (back)** (Codierung ØØ1): Positionierung auf den gerade bearbeiteten Satz des Datenbestandes

**F (first)** (Codierung Ø1Ø): Positionierung auf den 1. Satz des Datenbestandes

**L (last)** (Codierung Ø11): Positionierung auf den letzten aufgezeichneten Satz im Datenbestand (nur bei FD)

E (extented): Positionierung auf den nächsten freien Satz
(Codierung111) im Datentestand

Prüfungen:

bei READ
Ende der Datei erreicht (EOF)
Bereichsüberschreitung

bei WRITE
Schreibschutz
Bereichsüberschreitung

Belegung des z 2-Bytes:

| 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 |
|---|---|---|---|---|---|---|---|
| $I_1$ | $I_2$ | Ø | pos | | | M | W |

$I_n$ = Ø (n = 1, 2), d.h. n-te Adresse im Befehl direkt
$I_n$ = 1, d.h. n-te Adresse im Befehl indirekt
W = Ø, d.h. Warten auf Ausführung des Befehls
W = 1, d.h. kein Warten auf Ausführung des Befehls;
WAIT = Befehl muß folgen
M = Ø, d.h. Schreiben von Daten
M = 1, Schreiben "Markierter Satz"

Assemblerschreibweise für par:

W ohne WAIT (W = 1)
M Marked (M = 1)
WM (W = 1, M = 1)

| Namensfeld | Operationsfeld | Operanden- und Kommentarfeld | Read directly/ Write directly |
|---|---|---|---|
| | READD<br>WRITD | adr 1, adr 2, adr 3 [,par] | READD/<br>WRITD |

**Befehlsformat:**

41

| 45 | z 3 | adr 1 | adr 2 | adr 3 |

Anzahl I-Bit:

$3_Z$

**Funktion:**

Lesen/Schreiben eines direkt adressierten Satzes auf den Datenträger

**Bedingungscode:**

-

Die Operation steht unter Kontrolle des Dateidefinitionsblocks DDB, dessen Anfangsadresse durch adr 1 im Befehl spezifiziert ist.

Die Positionierung des jeweils zu lesenden/schreibenden Satzes erfolgt durch die Angabe der Satzadresse, die unter der Adresse adr 3 im Speicher zu finden ist. Die Satzadresse wird binär angegeben. Durch wahlfreie Angabe kann jede aktuelle Satzadresse innerhalb der Datei ausgewählt werden.

**Prüfungen**

bei READD
Markierter Satz gelesen
Bereichsüberschreitung

bei WRITD
Schreibschutz
Bereichsüberschreitung

Der auf dem Datenträger adressierte Satz wird in/aus den durch adr 2 gekennzeichneten Bereich gelesen/geschrieben. Die Adressierung des Satzes erfolgt über adr 3. Im Fehlerfall wird die aktuelle Satzadresse auf den fehlerhaften Satz eingestellt. Bei Satzadresse = Ø, =-1, =-2, =-3 wird der Dateikennsatz des ersten, zweiten, dritten, vierten im DDB spezifizierten Geräts angesprochen.

Belegung des x3-Bytes:

| 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 |
|---|---|---|---|---|---|---|---|
| $I_1$ | $I_2$ | $I_3$ | Ø | Ø | Ø | M | W |

$I_n$ = Ø (n = 1, 2, 3), d.h. n-te Adresse im Befehl direkt
$I_n$ = 1 , d.h. n-te Adresse im Befehl indirekt

M, W s. READ-Befehl

| Namens-feld | Operations-feld | Operanden- und Kommentarfeld | Read and search |
|---|---|---|---|
| | READS | adr 1, adr 2, pos [,par] | READS |

**Befehlsformat:**

43 | z 1 | adr 1 | adr 2 |

**Anzahl I-Bit:**

$2_z$

**Funktion:**

Suchen eines Satzes auf dem Datenträger entsprechend einer Suchmaske

**Bedingungscode:**

-

Die Operation läuft unter Einbeziehung eines DDB ab, dessen Anfangsadresse adr 1 im Befehl verankert ist.

Die Satzpositionierung wird analog dem Befehl READ durchgeführt. Für geblockte Dateien ist nur READ F (first) anwendbar.

**Prüfungen:**

Ende der Datei erreicht

Der Befehl bewirkt ein sequentielles Durchsuchen der Sätze der Datei, beginnend mit der durch pos spezifizierten Satzadresse entsprechend einer Suchmaske. Die Suchmaske wird ab adr 2 im E/A-Bereich bereitgestellt. Sie enthält an den entsprechenden Positionen die Codierungen, die verglichen werden sollen.
Die anderen Positionen enthalten NUL. Das Suchen erfolgt solange, bis ein Satz gefunden wird, der gleich der angegebenen Suchmaske ist. Markierte Sätze werden nicht mit verglichen. Der Satz, der der Suchmaske entspricht, wird in den

E/A-Bereich ab adr 2 geschrieben und seine Adresse als aktuelle Adresse in den DCB übernommen.

Wird kein Satz gefunden, der der angegebenen Suchmaske entspricht, wird zu EOF verzweigt.

Belegung des 2. 1-Bytes:

| 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 |
|---|---|---|---|---|---|---|---|
| $I_1$ | $I_2$ | Ø | pos | | | M | W |

analog READ-Befehl

Beispiel:

```
A N N N N 1 2 3 4 N N N . . .      Suchmaske
  U U U U       U U U
  L L L L       L L L
A 3 5 7 Ø 1 2 3 4 S C H . . .      Satz, der der Such-
                                   maske entspricht
```

# WRITE

**Schreiben sequentiell**

**write adr1, adr2, pos** **2 z-bit**

**44 | z2 | ad | r1 | ad | r2**

Aufbau des z 4-Bytes:

| 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 |
|---|---|---|---|---|---|---|---|
| $I_1$ | $I_2$ | $I_3$ | Ø | Ø | Ø | Ø | W |

$I_n$ = Ø, d.h. n-te Adresse im Befehl direkt
$I_n$ = 1, d.h. n-te Adresse im Befehl indirekt
W = Ø, d.h. Warten auf Ausführung
W = 1, d.h. kein Warten auf Ausführung

Prüfung:

Schlüsselindizierter Zugriff spezifiziert

| Namens-feld | Operations-feld | Operanden- und Kommenterfeld |
|---|---|---|
| | ONF | onfc, adr 1, adr 2 |

On file

ONF

**Befehlsformat:**

47 , z 3, adr 1 , adr 2

Anzahl I-Bit:

$2_Z$

**Funktion:**

Herstellen der Verbindung zwischen einem Unterbrechungsprogramm (adr 2) und der Datei (adr 1)

**Bedingungscode:**

-

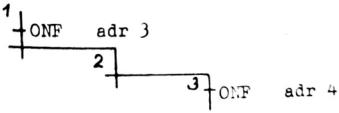
**Fehleranmeldung**

Die unter adr 2 angegebene Adresse wird zusammen mit der Bedingung onfc in eine systemeigene Tabelle eingetragen.

Tritt eine für die adr 1 angegebene Bedingung während der Abarbeitung eines E/A-Befehls auf, so wird in das Unterbrechungsprogramm (adr 2) verzweigt.

Ein ONF-Befehl gilt in der programmierten und allen nachfolgenden Ebenen solange, bis ein neuer ONF-Befehl spezifiziert wird. Bei Arbeit mit mehreren Programmebenen wird analog zu den erreichten Programmebenen eine Hierarchie der Fehleranmeldungen aufgebaut und verwaltet.
Bei Rückkehr von einer niederen in eine höhere Ebene tritt wieder die für diese Ebene gültige Anmeldung in Kraft.

**Beispiel:**



- Bedingung ON? mit adr 3 gilt in den Ebenen 1 und 2
- in der Ebene 3 gilt bis zu ON? mit adr 4 noch die vorhergehende Bedingung, erst ab ON? adr 4 dann diese Bedingung
- nach verlassen der Ebene 3 (mit RET in Ebene 2 zurück) gilt wieder ON? mit adr 3

Die erneute Programmierung eines ON?-Befehls mit der gleichen Bedingung in der gleichen Programmebene überschreibt die vorhergehende Anweisung.
Durch einen ON?-Befehl mit adr 2 = 0 wird eine eventuell bestehende Programmverbindung aufgehoben. In höheren Programmebenen bestehende Verbindungen werden dann wieder wirksam.
Soll eine ON?-Verzweigung während des Programmlaufs generell verhindert werden, so muß über adr 2 eine Routine angesprungen werden, die ausschließlich aus dem Befehl RET besteht.

Fehlerbehandlung

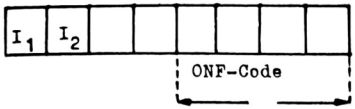
Eine ON?-Verzweigung ist einer Unterprogrammsprung äquivalent.

Durch den Befehl RET kann das unterbrochene Programm nach dem letzten eingestellten Makrobefehl fortgesetzt werden. Durch den Befehl EXIT wird die Rückkehrverbindung aufgehoben, und die Weiterarbeit erfolgt mit dem nach EXIT folgenden Befehl logisch in der Programmebene, in der der ON?-Befehl eingestellt wurde (Fehlermeldungen niederer Ebenen sind dann gelöscht).

ON?-Code (4 Bit):

| ON?-Code | | code (Assembler-Schreibweise) |
|---|---|---|
| 0 | Error | ERR |
| 1 | End of file | EOF |
| 2 | End of volume | EOV |
| 3 | Parity | PRY |
| 4 | Data transmission | DT |

Aufbau des z 3-Bytes:




Besitzt adr 1 den Wert Ø ... 255, so wird dieser als logischer Gerätename interpretiert. Das Betriebssystem SIOS läßt für logische Adressen nur den ONF-Code Ø, 1, 2, 3 zu.

In Verbindung mit einer Dateiadresse kann der ONF-Code einen Wert Ø ≤ ONF-Code ≤ 15 annehmen.
Bei MB-Arbeit muß adr 1 die logische Geräteadresse (H'Ø8') sein.

Besondere Funktion beim ONF-Code DT:

Im Unterschied zu den sonstigen ONF-Funktionen führt die Angabe dieses Befehls nach einem beliebigen Makrobefehl zur Verzweigung in das DFÜ-Unterbrechungsprogramm, sofern eine DFÜ-Aktivität (Empfang oder Senden) abgeschlossen wurde. Dadurch ist die simultane Bedienung der DFÜ zu anderen Aufgaben durch das Anwenderprogramm möglich.

| Namensfeld | Operationsfeld | Operanden- und Kommentarfeld |
|---|---|---|
| | WAIT<br>WAITC | adr<br>adr 1, adr 2 |

Weit/Wait and check

WAIT/
WAITC

Befehlsformat:

| 6 E | adr |
|---|---|
| 6 D | adr 1 | adr 2 |

Anzahl I-Bit:

1

Funktion:

Warten auf Beendigung einer E/A-Operation/und Testen der fehlerfreien Ausführung

Bedingungscode:

-

adr/adr 1 ist die Anfangsadresse des Dateidefinitionsblocks DDB.

Besitzt adr/adr 1 den Wert Ø ... 255, so wird dieser als logischer Gerätename interpretiert und die WAIT-Funktion bzgl. des physischen Geräts ausgeführt.

Die Programmfortsetzung wird bis zum Abschluß der letzten E/A-Operation auf die benannte Datei (physisches Gerät) angehalten.

Nach Verlassen des WAIT-Befehls ist im Statusselektorbyte der Status- und Fehlercode abgelegt.

Beim Befehl WAITC wird nach einem E/A-Fehler (Statusbit 7 gesetzt) zur Adresse adr 2 verzweigt. Das geschieht unabhängig davon, ob eine ERR-Adresse im DDB angegeben ist oder vorher ein entsprechender ONF-Befehl ausgeführt wurde.
Der Befehl wird ignoriert, wenn auf der benannten Datei (physisches Gerät) keine E/A-Operation angestoßen wurde.

Bei Arbeit mit dem 1/2"-Magnetband muß als adr/adr 1 die logische Geräteadresse H'Ø8' angegeben werden.

## 8.16. Ausführungszeiten von Makrobefehlen (MABS 1520)

### a) Befehle mit festen Ausführungszeiten

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| AB | 1,1 ms | BCT | 0,8 ms | GUIDE | 0,6 ms | ON | 0,8 ms |
| ABI | 1,0 ms | CALC | 0,9 ms | EXIT | 0,1 ms | ONF | 1,0 ms |
| ADRD | 0,9 ms | CALL | 0,8 ms | MB | 1,8 ms | PA | 1,0 ms |
| ADRE | 0,9 ms | CB | 1,0 ms | MLF | 0,6 ms | RESK | 0,7 ms |
| BC | 0,7 ms | CBI | 0,8 ms | MLS | 0,6 ms | RESS | 0,7 ms |
| BCA | 0,7 ms | CLI | 0,8 ms | MVA | 0,8 ms | RET | 0,4 ms |
| BCN | 0,7 ms | CLOCK | 1,4 ms | MVN | 0,8 ms | SB | 1,1 ms |
| BCK | 0,7 ms | DATE | 1,4 ms | MVBI | 0,9 ms | SETC | 1,3 ms |
| BCS | 0,7 ms | EX | 0,6 ms | MVI | 0,9 ms | SETD | 1,3 ms |
| BCKR | 0,7 ms | EOI | 0,4 ms | NOP | 0,4 ms | SETK | 0,7 ms |
| TM | 0,85ms | TIMER | 1,3 ms | SETS | 0,7 ms | SETPS | 0,7 ms |

### b) Befehle, bei denen die Ausführungszeiten von den Längenangaben und Zeichenkettenwerten abhängig sind

| | | |
|---|---|---|
| A<br>S | (z.B. 11 = 4 ; 12 = 3) | 3,3 ms |
| M | (Beisp. 1 bei der Befehlsbeschreibung) | 16,0 ms |
| D | (Beisp. 1 bei der Befehlsbeschreibung) | 20,0 ms |
| N | (z.B. 1 = 2) | 1,1 ms |
| O | (z.B. 1 = 1) | 1,0 ms |
| X | (z.B. 1 = 16) | 1,8 ms |
| MV | (Beisp. bei der Befehlsbeschreibung) | 1,1 ms |
| MVC | (z.B. 1 = 4) | 1,0 ms |
| MVM | (Beisp. bei der Befehlsbeschreibung) | 1,2 ms |
| C | (Beisp. bei der Befehlsbeschreibung) | 6,0 ms |
| CLC | (Beisp. bei der Befehlsbeschreibung) | 0,9 ms |
| CV | (Beisp. bei der Befehlsbeschreibung) | 1,3 ms |
| CVC | (Beisp. bei der Befehlsbeschreibung) | 1,5 ms |
| TR | (Beisp. bei der Befehlsbeschreibung) | 1,2 ms |
| PL | (Beisp. bei der Befehlsbeschreibung) | 1,0 ms |

### c) Restliche Befehle

E/A-Befehle sind abhängig von der Übertragungsgeschwindigkeit des entsprechenden externen Gerätes.

Anlage 1

Gruppeneinteilung der Makrobefehle nach der Operandenstruktur

| | | |
|---|---|---|
| 1. | BPC | RET, SOP, EOP, NOP, OFF, EOI, EXIT |
| 2. | OPC at 1 [,W] | ENTM, ENT, ENTS |
| 3. | OPS at 2 | CALL, CALC, OPEN, CLOSE, VLP, VLS, MVA, MVN, WAIT, GUIDE |
| 4. | OPC at 2 (OPC 2 Byte lang) | SETC, CLOCK, SETD, DATE, ADRD, ADRE, CALP |
| 5. | OPC at 2 Bedingungscodetyte wird aus Befehlsmnemonik generiert | B, BZ, BM, BP, BO, BNZ, BNM, BNP, BNO, BE, BL, BH, BNE, BNL, BNH |
| 6. | OPC at 1, at 1 | C, MV, MVM A, S |
| 7. | OPC at 1, at 2 | MVC, N, O, X, TR, CLC, CVC |
| 8. | OPC at 1, at 5 [,W] | ENTD |
| 9. | OPC at 1, at 1 | CV |
| 10. | OPC at 2, at 2 | AB, SB, MB, CB, BCT, EX, TIMER, WAITC |
| 11. | OPC at 2, at 3 | CVB |
| 12. | OPC at 3, at 2 | CVD |
| 13. | OPC at 3, at 3 | AP, APR, SP, SPR, C |
| 14. | OPC at 3, at 4 | PACK |
| 15. | OPC at 2, d 1 | MVI, CLI, TM |
| 16. | OPC at 2, d 2 | MVEI, API, CEI |
| 17. | OPC d 1 | SETPS |
| 18. | OPC 1 15 | SETS, SETK, RESS, RESK |
| 19. | OPC n, at 2 | BC |
| 20. | OPC n, at 3 | SL, SR |

Anlage 1:

| | | | |
|---|---|---|---|
| 21. | OPC | n, at 3 [,W] | ENTP |
| 22. | OPC | g, at 1 [,W] | GET |
| 23. | OPC | 1, at 2 | PA, PL |
| 24. | OPC | nz, at 1 [,W] | ENTF, ENTU |
| 25. | OPC | s 15, at 2 | BCS, BCK, BCKR |
| 26. | OPC | onc, at 2 | ON |
| 27. | OPC | stc, at 2 | BCN, BCA |
| 28. | OPC | n, at 1, at 1 | M, D |
| 29. | OPC | n, at 3, at 3 | MP, DP |
| 30. | OPC | n, at 4, at 3 | UNPK |
| 31. | OPC | g, at 1, at 5, [,W] | PUT |
| 32. | OPC | cnfc, at 2, at 2 | ONF |
| 33. | OPC | at 2, at 2, pos [,par] | READ, WRITE, READS |
| 34. | OPC | at 2, at 2, at 2 [,par] | READD, WRITD, READK |
| 35. | OPC | at 2, fctc [,W] | CTRL |
| 36. | OPC | g, nz, at 1 [,W] | PUTF |
| 37. | OPC | g, m, n, at 3 [,W] | PUTP |
| 38. | OPC | g [(c) ,pparlist] [,W] | POS |
| 39. | OPC | g, at 2 | CD |

Anlage 2

Makrobefehle (Blätter 1 - 11)

| | Bezeichnung | Assemblerschreibweise | Anzahl I-Bit | Codierung |
|---|---|---|---|---|
| A | Addition | A adr 1 [(l 1)] , adr 2 [(l 2)] | 2 | 10 , l 1, adr 1 , l 2 , adr 2 , |
| S | Subtraktion | S adr 1 [(l 1)] , adr 2 [(l 2)] | 2 | 11 , l 1, adr 1 , l 2 , adr 2 , |
| M | Multiplikation | M n, adr 1 [(l 1)] , adr 2 [(l 2)] | 2 | 12 , l 1, adr 1 , l 2 , adr 2 , n, |
| D | Division | D n, adr 1 [(l 1)] , adr 2 [(l 2)] | 2 | 13 , l 1, adr 1 , l 2 , adr 2 , n, |
| N | Logisches "UND" | N adr 1 [(l)] , adr 2 | 2 | 19 , l , adr 1 , adr 2 , |
| O | Logisches "ODER" | O adr 1 [(l)] , adr 2 | 2 | 1 A , l , adr 1 , adr 2 , |
| X | Logisches "Exclusive ODER" | X adr 1 [(l)] , adr 2 | 2 | 1 B , l , adr 1 , adr 2 , |
| AP | Addition gepackt | AP adr 1 [(k 1)] , adr 2 [(k 2)] | 2 | 3 A , p 1 , adr 1 , adr 2 , |
| APR | Subtraktion gepackt bei gesetztem "KØ-Bit" | APR adr 1 [(k 1)] , adr 2 [(k 2)] | 2 | 3 A , p 2 , adr 1 , adr 2 , |
| SP | Subtraktion gepackt | SP adr 1 [(k 1)] , adr 2 [(k 2)] | 2 | 3 A , p 3 , adr 1 , adr 2 , |
| SPR | Addition gepackt bei gesetzten "KØ-Bit" | SPR adr 1 [(k 1)] , adr 2 [(k 2)] | 2 | 3 A , p 4 , adr 1 , adr 2 , |
| MP | Multiplikation gepackt | MP n, adr 1 [(k 1)] , adr 2 [(k 2)] | 2 | 3 A , p 5 , adr 1 , adr 2 , |
| D | Division gepackt | DP n, adr 1 [(k 1)] , adr 2 [(k 2)] | 2 | 3 A , p 6 , adr 1 , adr 2 , |

Anlage 2, Blatt 2

| | Bezeichnung | Assemblerschreibweise | Anzahl I-Bit | Codierung |
|---|---|---|---|---|
| CP | Vorzeichengerechter Vergleich gepackt | CP adr 1 [k 1] , adr 2 [k 2] | 2 | 3 A \| p 7 \| adr 1 \| adr 2 |
| SL | Linksverschiebung gepackt | SL n, adr [(k)] | 1 | 3 C \| p 8 \| adr |
| SR | Rechtsverschiebung gepackt | SR n, adr [(k)] | 1 | 3 C \| p 9 \| adr |
| AB | Binäre Addition | AB adr 1, adr 2 | 2 | 30 \| adr 1 \| adr 2 |
| SB | Binäre Subtraktion | SB adr 1, adr 2 | 2 | 31 \| adr 1 \| adr 2 |
| MB | Binäre Multiplikation | MB adr 1, adr 2 | 2 | 32 \| adr 1 \| adr 2 |
| ABI | Addition eines 2-Byte-Direktop. | ABI adr, d 2 | 1 | 65 \| d \| 2 \| adr |
| MVBI | Transport eines 2-Byte-Direktop. | MVBI adr, d 2 | 1 | 64 \| d \| 2 \| adr |
| CB | Binärer Vergleich | CB adr 1, adr 2 | 2 | 33 \| adr 1 \| adr 2 |
| CBI | Binärer Vergleich mit 2-Byte-Direktop. | CBI adr, d 2 | 1 | 63 \| d \| 2 \| adr |

Anlage 2, Blatt 3

| | Bezeichnung | Assemblerschreibweise | An-zahl I-Bit | Codierung |
|---|---|---|---|---|
| MV | Transport rechts beginnend mit Auffüllen von NUL bis adr 1 | MV adr 1 [(l 1)], adr 2 [(l 2)] | 2 | 17, l 1, adr 1, l 2, adr 2, |
| MVC | Transport links beginnend | MVC adr 1 [(l)], adr 2 | 2 | 1 D, l, adr 1, adr 2, |
| MVM | Transport mehrfach links beginnend | MVM adr 1 [(l 1)], adr 2 [(l 2)] | 2 | 16, l 1, adr 1, l 2, adr 2, |
| MVI | Transport Direktoperand | MVI adr, d 1 | 1 | 6Ø, d 1, adr, |
| C | Numerischer Vergleich 2er maskierter numerischer Operanden | C adr 1 [(l 1)], adr 2 [(l 2)] | 2 | 14, l 1, adr 1, l 2, adr 2, |
| CLC | Logischer Vergleich links beginnend | CLC adr 1 [(l)], adr 2 | 2 | 1.C, l, adr 1, adr 2, |
| CLI | Vergleich mit Direktoperand | CLI adr 1, d 1 | 1 | 61, d 1, adr, |
| TM | Prüfen der Bitkonfiguration eines Bytes | TM adr 1, d 1 | 1 | 62, d 1, adr, |

| | Bezeichnung | Assemblerschreibweise | Anzahl I-Bit | Codierung |
|---|---|---|---|---|
| CV | Konvertieren Dezimalzahl in Binärzahl | CV adr 1, adr 2 [(1)] | 2 | 34 , 1 , adr 1 , adr 2 , |
| CVC | Konvertieren Binärzahl in Dezimalzahl | CVC adr 1 [(1)] , adr 2 | 2 | 35 , 1 , adr 1 , adr 2 , |
| CVB | Konvertieren gepackte Dezimalzahl in Binärzahl | CVB adr 1, adr 2 [(k)] | 2 | 38 , p 11 , adr 1 , adr 2 , |
| CVD | Konvertieren Binärzahl in gepackte Dezimalzahl | CVD adr 1 [(k)] , adr 2 | 2 | 39 , p 11 , adr 1 , adr 2 , |
| PACK | Konvertieren ungepackte Dezimalzahl in gepackte Dezimalzahl | PACK adr 1 [(k)], adr 2 [(1)] | 2 | 37 , p 12 , adr 1 , adr 2 , |
| UNPK | Konvertieren gepackte Dezimalzahl in ungepackte Dezimalzahl | UNPK m, adr 1 [(1)], adr 2 [(k)] | 2 | 36 , p 12 , adr 1 , adr 2 , m , |
| TR | Codewandlung | TR adr 1 [(1)] , adr 2 | 2 | 18 , 1 , adr 1 , adr 2 , |

Anlage 2, Blatt 5

| | Bezeichnung | Assemblerschreibweise | | Anzahl I-Bit | Codierung |
|---|---|---|---|---|---|
| B | Unbedingter Sprung | B | adr | 1 | Ø D, F Ø, adr |
| BC | Bedingter Sprung in Abh. vom Bedingungscode | BC | n, adr | 1 | Ø D, n, Ø, adr |
| BCA | in Abh. vom Starttastencode Reg. A | BCA | stc, adr | 1 | Ø 3, stc, adr |
| BCN | in Abh. vom Starttastencode Reg. N | BCN | stc, adr | 1 | Ø 4, stc, adr |
| BCK | in Abh. vom Sel.-register K | BCK | s 15, adr | 1 | Ø 6, sc, adr |
| BCKR | in Abh. vom Sel.-register KR | BCKR | s 15, adr | 1 | Ø 7, sc, adr |
| BCS | in Abh. vom Statusreg. S | BCS | s 15, adr | 1 | Ø 5, sc, adr |
| BCT | in Abh. von einem Zählwert | BCT | adr 1, adr 2 | 2 | 3 D, adr 1, adr 2 |
| TIMER | in Abh. von der Zeit | TIMER | adr 1, adr 2 | 2 | 3 F, adr 1, adr 2 |
| ON | in Abh. vom ON-Code | ON | onc, adr | 1 | Ø 8, onc, adr |
| EXIT | Aufheben des Unterbrechungsstatus | EXIT | | Ø | CB |

Anlage 2, Blatt 6

| | Bezeichnung | Assemblerschreibweise | Anzahl I-Bit | Codierung |
|---|---|---|---|---|
| CALL | Aufruf eines UP in Makrobefehlssprache | CALL adr | 1 | [illegible] B, adr |
| CALC | Aufruf eines UP in Mikrobefehlssprache | CALC adr | 1 | [illegible] C, adr |
| CALP | Laden eines Progr. und Starten | CALP adr | 1 | [illegible], [illegible], adr |
| RET | Rückkehr vom UP | RET | [illegible] | [illegible] 1 |
| PA | Laden der Adr. der Parameterliste in das UP | PA 1, adr | 1 | [illegible], 1, adr |
| PL | Laden Parameterliste in das UP | PL 1, adr | 1 | [illegible] A, 1, adr |
| SOP | Programmende | SOP | [illegible] | [illegible] 2 |
| SOP | Start eines Makroprogramms | SOP | [illegible] | [illegible] |
| SOI | Aufheben Interpreterstatus | SOI | [illegible] | [illegible] |
| WTA | Transport Starttestenregister A | WTA adr | 1 | [illegible], adr |

| | Bezeichnung | Assemblerschreibweise | An-zahl I-Bit | Codierung |
|---|---|---|---|---|
| MVN | Transport Start-tastenregister N | MVN adr | 1 | 69 , adr , |
| MLF | Modifikation des 2. Byte des Folgebefehls | MLF adr | 1 | 66 , adr , |
| MLS | Modifikation des 5. Bytes des Folgebefehls | MLS adr | 1 | 67 , adr , |
| NOP | Keine Operation | NOP | Ø | ØØ , |
| OFF | Ausschalten des Geräts | OFF | Ø | CC , |
| SETK | Setzen Selektoren Gruppe K | SETK 115 | Ø | C 6 , 1 , 15 , |
| RESK | Rücksetzen Selektoren Gruppe K | RESK 115 | Ø | C 7 , 1 , 15 , |
| SETS | Setzen Selektoren Gruppe S | SETS 115 | Ø | C 4 , 1 , 15 , |
| RESS | Rücksetzen Selektoren Gruppe S | RESS 115 | Ø | C 5 , 1 , 15 , |
| GUIDE | Bedienerführung | GUIDE adr | 1 | 6 A , adr , |
| SETD | Eintragen Datum in Datumsregister | SETD adr | $1_z$ | CD , Ø5 , adr , |

| | Bezeichnung | Assemblerschreibweise | An-zahl I-Bit | Codierung |
|---|---|---|---|---|
| SETC | Eintragen Zeit in Zeitregister | SETC adr | $1_z$ | CD, Ø4, adr, |
| DATE | Bereitstellen des aktuellen Datums | DATE adr | $1_z$ | CD, Ø3, adr, |
| CLOCK | Bereitstellen der aktuellen Zeit | CLOCK adr | $1_z$ | CD, Ø2, adr, |
| ADRD | Bereitstellen Anfangsadresse Bildwiederholspeicher | ADRD adr | $1_z$ | CD, Ø1, adr, |
| ADRE | Bereitstellen Endadresse RAM | ADRE adr | $1_z$ | CD, ØØ, adr, |
| EX | Befehlsaustausch mit Modifizierung | EX adr 1, adr 2 | 2 | 3 E, adr 1, adr 2, |
| SETPS | Setzen des Programmstatus | SETPS d1 | Ø | CA, d 1, |
| ENT | Numerische Eingabe | ENT adr [(1)] [,W] | 1 | 24, 1, adr, |
| ENTP | Eingabe gepackt mit Speicherung | ENTP n, adr [(k)] [,W] | 1 | 21, p 1Ø, adr, |
| ENTM | Zeicheneingabe mit Speicherung | ENTM adr [(1)] [,W] | 1 | 25, 1, adr, |
| ENTF | Mehrzeilige Zeicheneingabe | ENTF nz, adr [(1)] [,W] | 1 | 23, 1, adr, nz, |
| ENTS | Zeicheneingabe in d. Speicher mit automatischen Abbruch bei erreichter Zeichenanzahl. | ENTS adr [(1)] [,W] | 1 | 22, 1, adr, |

| | Bezeichnung | Assemblersprachformat | Anzahl I-Bit | Codierung |
|---|---|---|---|---|
| NNTD | Numerische Eingabe mit Meldungsanzeige und Speicherung | NNTD adr 1 [(1)], adr 2 [,W] | 1 | 20, 1, adr 1, adr 2, |
| NNTU | Zeichenumsetzung auf dem Meldegerät | NNTU ns, adr [(1)][,W] | 1 | 26, 1, adr, ns, |
| PUT | Ausgabe numerischer Daten | PUT g, adr 1 [(1)], adr 2 [,W] | 1 | 28, 1, adr 1, g, adr 2, |
| PUTC | Zeichenausgabe | PUTC g, adr [(1)][,W] | 1 | 2 A, 1, adr, g, |
| PUTP | Ausgabe gepackt mit Maskenaufbereitung | PUTP g, m, n, adr [(k)][,W] | 1 | 29, p 10, adr, g, m, |
| PUTP | Mehrzeilige Zeichenausgabe | PUTP g, ns, adr [(1)][,W] | 1 | 2 E, 1, adr, g, ns, |
| GET | Eingabe vom logischen Gerät | GET g, adr [(1)][,W] | 1 | 2 , 1, adr, g, |
| POS | Positionierung Drucker bzw. Bildschirm | POS g [(c)] [,pparlist] [,W] | 0 | 2 C, par, H,V,g, |
| CD | Definition einer Steuertelle | CD g, adr | 1 | 27, par, adr, g, |
| CTRL | Steuerung peripherer Geräte | CTRL adr, fctc [,W] | 1 | 2 E, fctc, adr, |

Anlage 2, Blatt 10

| | Bezeichnung | Assemblerschreibweise | Anzahl I4Bit | Codierung |
|---|---|---|---|---|
| OPEN | Eröffnen einer Datei | OPEN adr | 1 | 6 B, adr |
| CLOSE | Abschließen einer Datei | CLOSE adr | 1 | 6 C, adr |
| READ | Lesen eines Satzes vom Datenträger | READ adr 1, adr 2, pos [,par] | $2_z$ | 40, z 2, adr 1, adr 2 |
| WRITE | Schreiben eines Satzes auf den DT | WRITE adr 1, adr 2, pos [,par] | $2_z$ | 44, z 2, adr 1, adr 2 |
| READD | Lesen eines direkt adressierten Satzes | READD adr 1, adr 2, adr 3 [,par] | $3_z$ | 41, z 3, adr 1, adr 2, adr 3 |
| WRITD | Schreiben eines direkt adressierten Satzes | WRITD adr 1, adr 2, adr 3 [,par] | $3_z$ | 45, z 3, adr 1, adr 2, adr 3 |
| READS | Suchen eines Satzes entsprechend einer Suchmaske | READS adr 1, adr 2, pos [,par] | $2_z$ | 43, z 1, adr 1, adr 2 |
| READK | Schlüsselindiziertes Lesen eines Satzes | READK adr 1, adr 2, adr 3 [,par] | $3_z$ | 42, z 4, adr 1, adr 2, adr 3 |

Anlage 2, Blatt 11

| | Bezeichnung | Assemblerschreibweise | An-zahl I-Bit | Codierung |
|---|---|---|---|---|
| ONF | Herstellen einer Verbindung zwischen UP und Datei | ONF onfc, adr 1, adr 2 | $2_z$ | 47 \| z 3 \| adr 1 \| adr 2 |
| WAIT | Warten auf Beendigung einer E/A-Op. | WAIT adr | 1 | 6 E \| adr |
| WAITC | Warten auf Beendigung einer E/A-Op. u. Testen der fehlerfreien Ausführung | WAITC adr 1, adr 2 | 1 | 6 D \| adr 1 \| adr 2 |

Anlage 2, Blatt 12

Pseudobefehle

| | | | |
|---|---|---|---|
| Beginn Quellprogramm | | PN | symbol |
| Ende Quellprogramm | | END | |
| Seitenschaltung Übersetzungsprotokoll | | EJEC | |
| Zeilenvorschub Übersetzungsprotokoll | | SPACE | t |
| Kennzeichnung externer Symbole | | EXTRN | symtollist |
| Definition symbolischer Eintrittpunkte | | ENTRY | symtollist |
| Speicherbereichsdefinition | [symtol:] | DS | adr |
| Erweiterte Speicherbereichsdefinition | [symbol:] | DS | adr * adr |
| Konstantendefinition | [symtol:] | DC | wert |
| . Typ F | [symtol:] | DC | F'f' |
| . Typ W | [symbol:] | DC | W'q' |
| . Typ D | [symbol:] | DC | D'r' |
| . Typ CL | [symbol:] | DC | CL l'$c_1$ ... $c_l$' |
| . Typ C | [symtol:] | DC | C'$c_1$ ... $c_l$ |
| . Typ H | [symtol:] | DC | H'$h_1$ ... $h_i$ |
| . Typ NL | [symtol:] | DC | NL $\bar{c}$'s' |
| . Typ N | [symtol:] | DC | N'r' |
| . Typ A | [symbol:] | DC | A(adr) |
| . Typ Y | [symtol:] | DC | Y(adr) |
| . Typ X | [symtol:] | DC | $zkx_4$ |
| . Typ I | [symtol:] | DC | I'$\bar{I}$' |
| . Typ V | [symbol:] | DC | V(symtol) |
| Wertzuweisung für ein Symbol | symbol: | EQU | adr |

Anlage 2, Blatt 13

Pseudobefehle

| | | | |
|---|---|---|---|
| Setzen Speicher-Platzzuweisungs-zähler | | ORG | adr |
| Definition des statischen Datei-definitionsblocks | symbol: | DDB | ddtparlist |
| | ddtparlist | ::= | ddtparlist,ddtpar/ddtpar |
| | ddtpar ::= | | Schlüsselwort-Spezifikation |
| Definition des dynamischen Datei-definitionsblocks | symbol: | DDB | $\overline{I}$ |

Anlage 3, Blatt 1

| | | |
|---|---|---|
| Adreßdirektwert | nnnnn | nnnnn ::= $\left\{ \begin{array}{r} 0 \leq f \leq 65535 \\ X'0000' \leq zkX_4 \leq X'FFFF' \end{array} \right\}$ |
| Ausdrücke | adr | adr ::= $\left[ \left\{ \pm \right\} \text{term} \left[ \left\{ \pm \right\} \text{term} \right] \ldots \right.$ |
| Term | term | term ::= $\left\{ \begin{array}{l} \text{symbol} \\ \text{nnnnn} \\ zkh_4 \\ zkc_2 \\ * \end{array} \right\}$ |
| Direktoperand der Länge 1 Byte | d 1 | d 1 ::= at 5 |
| Direktoperand der Länge 2 Byte | d 2 | d 2 ::= at 5 |
| Absoluter Ausdruck mit eingeschränktem Wert | g | g ::= adr , Wert von adr $\leq$ 255 und absolut |
| Sprungbedingung in Abhängigkeit von Selektoren | s 15 | s 15 ::= $\left\{ \begin{array}{l} n \\ Nn \end{array} \right\}$ , N Negation |
| Bitpositionen des K-, S-, KR-Registers | l 15 | l 15 ::= $\left\{ \begin{array}{l} n \\ n, \text{ l } 15 \end{array} \right\}$ |

| ON-Codes | onc | onc ::= $\left\{\begin{matrix}E\\D\\Z\end{matrix}\right\}_p$ | ERROR<br>DIVISION<br>POWER ZERO-DIVISION |
|---|---|---|---|
| Alphanumerische Zeichen | $c_i$ | $c_i \in \{A, \ldots Z, Ø, 1, \ldots 9, @, \ldots\}$ | |
| Hexedezimalziffern | $h_i$ | $h_i \in \{Ø, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F\}$ | |
| Symbol | symbol | 1 ... 5 Zeichen, 1. Zeichen = Buchstabe, keine Sonderzeichen, kein Space ( ß )! | |
| Bezugnahme auf Speicherplatzzuweisungszähler | * | | |
| Zeichenketten | | | |
| Typ X | $zkx_i$ | $zkx_i$ ::= X'h[h]$*^{i-1}$'. , i = 2,4 | |
| Typ H | $zkh_i$ | $zkh_i$ ::= H'h[h]$*^{i-1}$'. | |
| Typ C | $skc_i$ | $zkc_i$ ::= C'c[c]$*^{i-1}$' | |
| Dezimalzahlen | c | $1 \leq c \leq 16$ | |
| | $\bar{c}$ | $1 \leq (\bar{c}) \leq 17$ | |
| | f | $Ø \leq f \leq 65535$ | |
| | k | $k \in \{4,8\}$ | |
| | l | $1 \leq l \leq 256$ | |
| | $\bar{l}$ | $Ø \leq (\bar{l}) \leq 255$ | |

| | | | | |
|---|---|---|---|---|
| | n | $0 \leq n \leq 15$ | | |
| | ns | $1 \leq ns \leq 255$ | | |
| | q | $\pm (0 \leq q \leq 10^{7} - 1)$ | | |
| | r | $\pm (0 \leq r \leq 10^{15} - 1)$ | | |
| | s | $\pm (0 \leq s \leq 10^{0-1} - 1)$ | | |
| | t | $0 \leq t$ 512 | | |
| OFF-Codes | onfc | onfc ::= | ERR | ERROR |
| | | | EOF | END OF FILE |
| | | | EOV | END OF VOLUME |
| | | | MRK | MARKED |
| | | | DT | DATA TRANSMISSION |
| Parameter für E/A-Operationsspezifikation | par | par ::= | W | |
| | | | M | |
| | | | WM | |
| | w | wie par ::= | W | |
| Positionierung bei Sequentiellen Label E/A-Befehlen | pos | pos ::= | F | FIRST |
| | | | N | NEXT |
| | | | B | BACK |
| | | | L | LAST |
| | | | E | EXIT |

| | | |
|---|---|---|
| Positionierparameterliste | pparlist | pparlist ::= pparlist, ppar/ppar<br>ppar ::= { H {+/-/=} g / V {+/-/=} g / VT / I = adr / FF / R } |
| Funktionscodes | fctc | fctc ::= RBC/WTM/FOR/REW/BSR/EBC/BSF/FSF/TRP/WON/WOF/RND/<br>/PUO /BWR/SPO/CFC /STS/GBO/CUT/FLF/ELF/KPN/KPI/SDC/EDC/NND/FL2/BL2/BSS/DEL/ |
| Sterntastencodes | stc | stc ::= Ø/1/ ... /63/ET1 /ET2 /ET/SEND/SKIP/REC/<br>PA1 /PA2 /PA3 /CLEAR/CNCL/ENTER/CL/CA/FV/FB |
| Maske | m | m ::= [int][deza] fs ds<br>int ::= NI/HP/HS/TP/TS<br>deza::= C/P<br>fs ::= Z/S/*<br>Ø ≤ ds ≤ 3 |

$$m ::= \left[\begin{Bmatrix} NI \\ HP \\ HS \\ TP \\ TS \end{Bmatrix}\right] \left[\begin{Bmatrix} C \\ P \end{Bmatrix}\right] \begin{Bmatrix} Z \\ S \\ * \end{Bmatrix} \begin{Bmatrix} \emptyset \\ 1 \\ 2 \\ 3 \end{Bmatrix}$$

Konstentenwert wert

$$wert ::= \left\{\begin{pmatrix} \text{F'f'} \\ \text{W'q'} \\ \text{D'r'} \\ \text{CLl'}c_1 \ldots c_1\text{'} \\ \begin{Bmatrix} \text{zkc}_1 \\ \text{C'}c_1 \ldots c_1\text{'} \end{Bmatrix} \\ \begin{Bmatrix} \text{zkh}_t \\ \text{H'}h_1 \ldots h_t\text{'} \end{Bmatrix} \\ \text{NL } \overline{c}\text{'s'} \\ \text{N'r'} \\ \begin{Bmatrix} \text{zkx}_4 \\ \text{X'}h_1h_2h_3h_4\text{'} \end{Bmatrix} \\ \text{I' } \overline{l}\text{'} \\ \text{V'symbol'} \\ \text{A'adr'} \end{pmatrix}\right\}$$

| Symbolliste | symbollist | symbollist ::= symbollist, symbol/symbol |
|---|---|---|
| DDB-Parameter | ddbpar | DCB = adr<br>NA = {symbol 6 / adr, 1 [,1]*4 / zkh$_8$}<br>DT = dt  dt ::= {R / RW}<br>SI = si  si ::= {COPY / SAVE / DEL / ERASE}<br>ORG = org$_{FD}$/  org$_{FD}$ ::= {KEYC (it, kp, kl, ,lit) / SEQ / DIR}<br>ORG = org$_{KMBG}$ ::= {SIM / BAS / COM}<br>PT = d 1<br>EOF = adr<br>ERR = adr<br>IOA = adr<br>IOS = adr<br>RCS = adr<br>DA = (lda)  lda ::= g [,g]* i - 1 |

## Anlage 4

### Systemanforderungen in bezug auf die automatische Bindefunktion für Programme/Unterprogramme

1. In allen Programmen/Unterprogrammen (Anwenderprogramme), die mit dem Binder aus SIOS 1520 verbunden werden sollen, gilt der Programmname als implizite Marke des ersten Befehls (erster Eintrittspunkt).

2. Existieren für ein aufzurufendes Programm neben o.g. Programmanfang weitere Eintrittspunkte, so müssen in der EXTRN-Anweisung des aufrufenden Programms Programmname und benötigte Eintrittspunkte aufgeführt werden.

Beispiel:

| Aufrufendes Programm | | Aufgerufenes Programm | |
|---|---|---|---|
| ⋮ | | PN | AB |
| CALL | AB | ⋮ | |
| ⋮ | | AB1 : ... | |
| CALL | AB1 | ⋮ | |
| ⋮ | | AB2 : ... | |
| CALL | AB3 | ⋮ | |
| ⋮ | | AB3 : ... | |
| EXTRN | AB, AB1 , AB3 | ⋮ | |
| | | ENTRY | AB1 , AB2, AB3 |

3. In einem aufrufbaren Programm sind in der ENTRY-Anweisung alle Eintrittspunkte anzugeben.

Beispiel: Es existiere ein Programm AB mit den Eintrittspunkten AB1 , AB2 , AB3 .

⋮

ENTRY AB1 , AB2 , AB3

4. Das "X" als erstes Zeichen des symbolischen Programmnamens ist den Standardunterprogrammen (POS-Lösungen), die mit dem System bereitgestellt werden können, vorbehalten. Der Anwender sollte Symbole vermeiden, die mit "X" beginnen.

[illegible]